Tkinter - Widgets

ICTACADEMY

Error/ Warning

Information

Flashback

TRY IT

Class Exercise

AGENDA

1. What Tkinter is and its importance in python GUI programming
2. Hierarchy of widgets and how they are organized
3. Main components of A Tkinter application: *root window, frames & widgets*
4. Widgets using GUI - drop down menus
5. Toolbar
6. Scale widget
7. Spinbox widget
8. Messagebox
9. Graphics and shapes-line graphics
10. Graphics and shapes - box
11. Graphics and shapes - canvas
12. Graphics and shapes - images in GUI

1. What Tkinter Is and Its Importance in Python GUI Programming

What Tkinter is and its importance in python GUI programming

Tkinter

- Tkinter is the standard GUI (Graphical User Interface) library for Python, providing a fast and easy way to create GUI applications.
- It is a thin object-oriented layer on top of the Tcl/Tk GUI toolkit.
- The name Tkinter comes from ***Tk inter***face.

What Tkinter is and its importance in python GUI programming

Importance

- Standard library integration
- Cross-platform compatibility
- Ease of use
- Rich widget set
- Event-driven programming
- Flexible layout management
- Customizable appearance
- Canvas widget
- Extensible
- Good documentation
- Integration with other libraries
- Stable

What Tkinter is and its importance in python GUI programming

Creating a Simple Window:

```python
import tkinter as tk

# Create the main window
root = tk.Tk()
root.title("Hello, this is a simple GUI program using Tkinter")
root.geometry("300x200")

# Run the main event loop
root.mainloop()
```

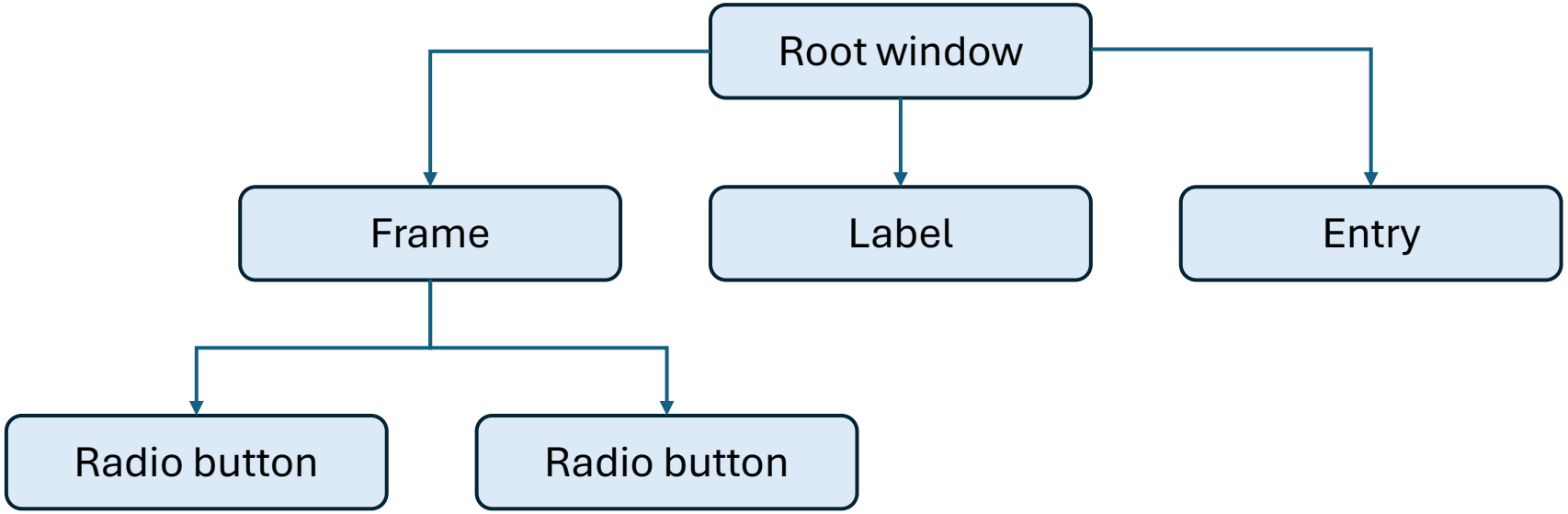
```
D:\Algoritmo\PythonGUI\projectHello\.venv\Scripts\python.exe D:\Algoritmo\PythonGUI\projectHello\hello.py
```

2. Hierarchy of Widgets and How They Are Organized

Hierarchy of widgets and how they are organized

Hierarchy of widgets

In Tkinter, widgets are organized in a hierarchical manner. The main window (or root window) is at the top of the hierarchy, and other widgets are added as children to this window or to other widgets.



Hierarchy of widgets and how they are organized

How they are organized

- In Tkinter, widgets are organized in a tree-like structure.
- The tree-like oraganization is essential in Tkinter because it defines the parent-child relationship between widgets, influencing their placement and behavior.
- Each widget can have only one parent, but a parent can have multiple child widgets.

Hierarchy of widgets and how they are organized

TRY IT

Create a Tkinter application with the following hierarchy

- **Root Window**: The main application window.
  - **Frame 1**: A frame inside the root window.
    - Label 1: A label inside Frame 1 with the text "Label in Frame 1".
    - Button 1: A button inside Frame 1 with the text "Button in Frame 1".
  - **Frame 2**: Another frame inside the root window.
    - Label 2: A label inside Frame 2 with the text "Label in Frame 2".
    - Button 2: A button inside Frame 2 with the text "Button in Frame 2".

3. Main Components of Tkinter Application:
root window, frames & widgets

Main components of Tkinter application: *root window, frames & widgets*

Main components of Tkinter application

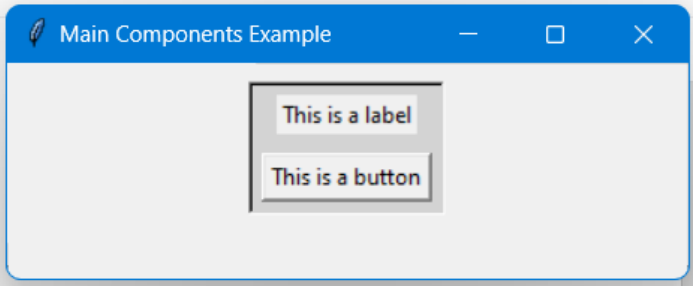
- **Root Window**: The main window of a Tkinter application. It is the parent container for all other widgets and frames.
- **Frames**: Containers used to organize and group widgets within the root window. They help in structuring the layout of the application.
- **Widgets**: The individual UI elements like buttons, labels, entry fields, etc., that are placed inside frames or directly within the root window.

Main components of Tkinter application: *root window, frames & widgets*

Example - Main components of Tkinter application

mainCompTkinter.py

```python
import tkinter as tk

# Create the root window
root = tk.Tk()
root.title("Main Components Example")

# Create a frame and add it to the root window
frame = tk.Frame(root, bg="lightgrey", bd=2, relief=tk.SUNKEN)
frame.pack(padx=10, pady=10)

# Create a label and add it to the frame
label = tk.Label(frame, text="This is a label")
label.pack(padx=5, pady=5)

# Create a button and add it to the frame
button = tk.Button(frame, text="This is a button")
button.pack(padx=5, pady=5)

# Start the main event loop
root.mainloop()
```



Main components of Tkinter application: *root window, frames & widgets*

ICTACADEMY

Explanation – Example of main components of Tkinter application

Root Window:

- Created using `tk.Tk()`
- The title is set using root.title("Main Components Example").

Frame:

- Created using `tk.Frame(root, ...)`
- Configured with background color (bg), border (bd), and relief style (relief).
- Added to the root window using frame.pack(padx=10, pady=10) which adds padding around the frame.

Label:

- Created using `tk.Label(frame, text="This is a label")`
- Added to the frame using label.pack(padx=5, pady=5) which adds padding around the label.

4. Widgets Using GUI - Dropdown Menus

Widgets using GUI - drop down menus

Drop-Down Menu

In Tkinter, a dropdown, often referred to as a "dropdown menu" or a "dropdown list," is a graphical control element that allows users to select one option from a list of predefined choices.

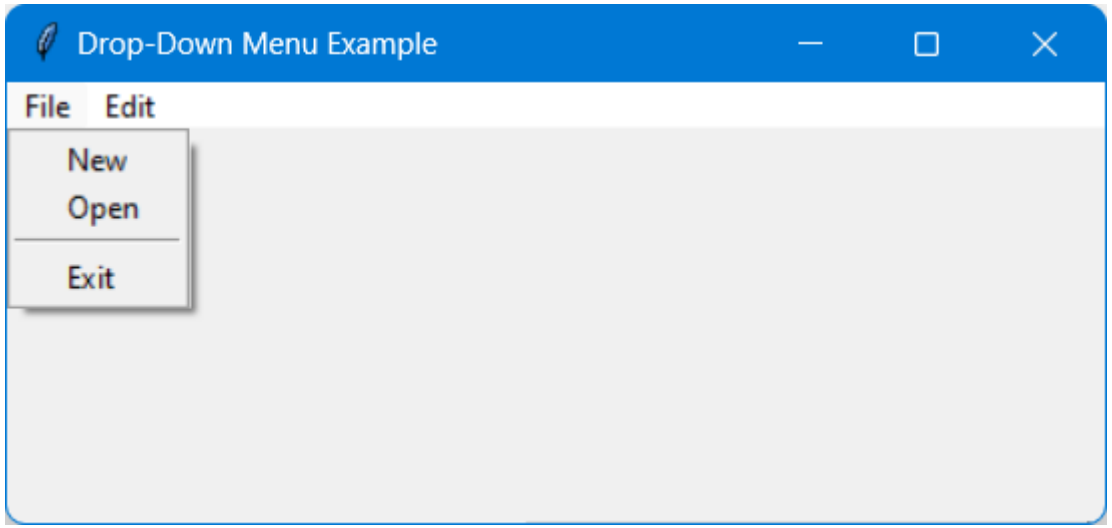
Steps to create a dropdown:

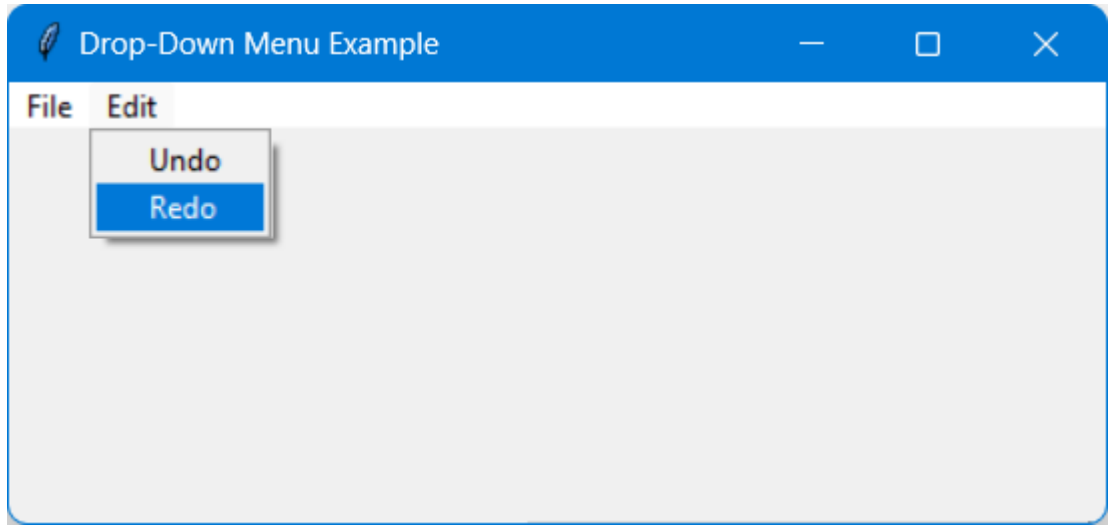
1. Create the Root Window: Initialize the main application window.
2. Create a Menu Widget: This will be the main menu bar.
3. Configure the Menu: Set the menu as the root window's menu.
4. Create Submenus: Add items to the menu bar and create submenus for drop-down functionality.
5. Add Commands to Submenus: Populate the submenus with commands or actions.

Widgets using GUI - drop down menus

Example – GUI of Drop-Down Menu

Let’s create a Tkinter application with a menu bar that contains two menus: "File" and "Edit". The "File" menu should have the options "New", "Open", and "Exit". The "Edit" menu should have the options "Undo" and "Redo". Display a message in the console when any menu item is clicked.





Widgets using GUI - drop down menus

Example – Code of Drop-Down Menu

dropdownMenu.py

```python
import tkinter as tk
from tkinter import messagebox

4 usages
def menu_callback(action):
    print(f"{action} menu item clicked")

# Create the root window
root = tk.Tk()
root.title("Drop-Down Menu Example")

# Create the main menu bar
menu_bar = tk.Menu(root)
root.config(menu=menu_bar)

# Create the File menu and add it to the menu bar
file_menu = tk.Menu(menu_bar, tearoff=0)
menu_bar.add_cascade(label="File", menu=file_menu)
```

```python
# Add commands to the File menu
file_menu.add_command(label="New", command=lambda: menu_callback("New"))
file_menu.add_command(label="Open", command=lambda: menu_callback("Open"))
file_menu.add_separator()
file_menu.add_command(label="Exit", command=root.quit)

# Create the Edit menu and add it to the menu bar
edit_menu = tk.Menu(menu_bar, tearoff=0)
menu_bar.add_cascade(label="Edit", menu=edit_menu)

# Add commands to the Edit menu
edit_menu.add_command(label="Undo", command=lambda: menu_callback("Undo"))
edit_menu.add_command(label="Redo", command=lambda: menu_callback("Redo"))

# Start the main event loop
root.mainloop()
```

Widgets using GUI - drop down menus

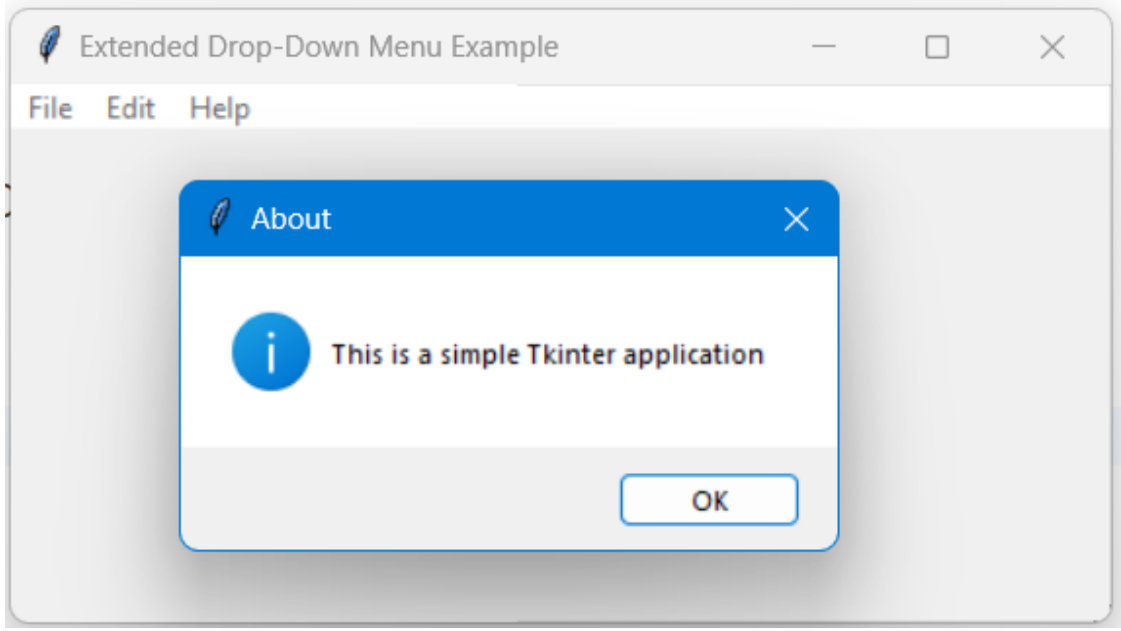
Explanation - Example of Drop-Down Menu

- Menu Bar: The main menu bar is created using `tk.Menu(root)`.
- Configuring Menu Bar: The menu bar is set as the root window's menu using `root.config(menu=menu_bar)`.
- File Menu:
  - Created using `tk.Menu(menu_bar, tearoff=0)`.
  - Added to the menu bar using `menu_bar.add_cascade(label="File", menu=file_menu)`.
  - Commands ("New", "Open", "Exit") are added to the file menu using `file_menu.add_command(...)`.
  - A separator is added using `file_menu.add_separator()`.
  - The "Exit" command is linked to `root.quit` to close the application.
- Edit Menu:
  - Created using `tk.Menu(menu_bar, tearoff=0)`.
  - Added to the menu bar using `menu_bar.add_cascade(label="Edit", menu=edit_menu)`.
  - Commands ("Undo", "Redo") are added to the edit menu using `edit_menu.add_command(...)`.

Widgets using GUI - drop down menus

TRY IT

Extend the previous example by adding a "Help" menu with an "About" option that shows an informational message box with the message "This is a simple Tkinter application".



5. Toolbar

ICTACADEMY

Toolbar

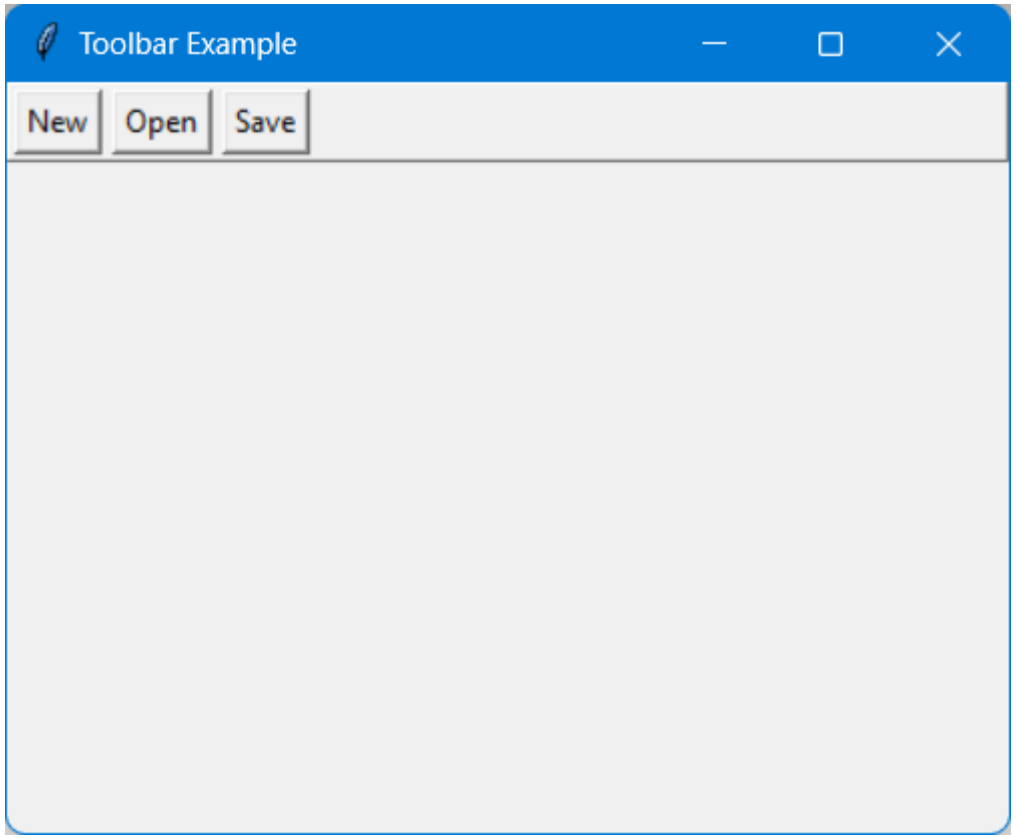
A toolbar in Tkinter can be created using the Frame widget, which acts as a container for buttons or other widgets that represent tools or actions. Toolbars are typically placed at the top of the application window and provide quick access to commonly used functions.

Steps to create a Toolbar:

1. Create the Root Window: Initialize the main application window.
2. Create a Frame for the Toolbar: This frame will hold the toolbar buttons.
3. Add Buttons to the Toolbar: Create and add buttons to the toolbar frame.
4. Pack the Toolbar Frame: Place the toolbar frame at the top of the root window.

Example – GUI of Toolbar

Create a Tkinter application with a toolbar that contains three buttons: "New", "Open", and "Save". Each button should print a message to the console when clicked.



Toolbar

Example – Code of Toolbar

toolbar.py

```python
import tkinter as tk

1 usage
def new_file():
    print("New file created")

1 usage
def open_file():
    print("File opened")

1 usage
def save_file():
    print("File saved")

# Create the root window
root = tk.Tk()
root.title("Toolbar Example")
root.geometry("400x300")
# Create a frame for the toolbar
toolbar = tk.Frame(root, bd=1, relief=tk.RAISED)
toolbar.pack(side=tk.TOP, fill=tk.X)
```

```python
# Create and add buttons to the toolbar
new_button = tk.Button(toolbar, text="New", command=new_file)
new_button.pack(side=tk.LEFT, padx=2, pady=2)

open_button = tk.Button(toolbar, text="Open", command=open_file)
open_button.pack(side=tk.LEFT, padx=2, pady=2)

save_button = tk.Button(toolbar, text="Save", command=save_file)
save_button.pack(side=tk.LEFT, padx=2, pady=2)

# Start the main event loop
root.mainloop()
```

Toolbar

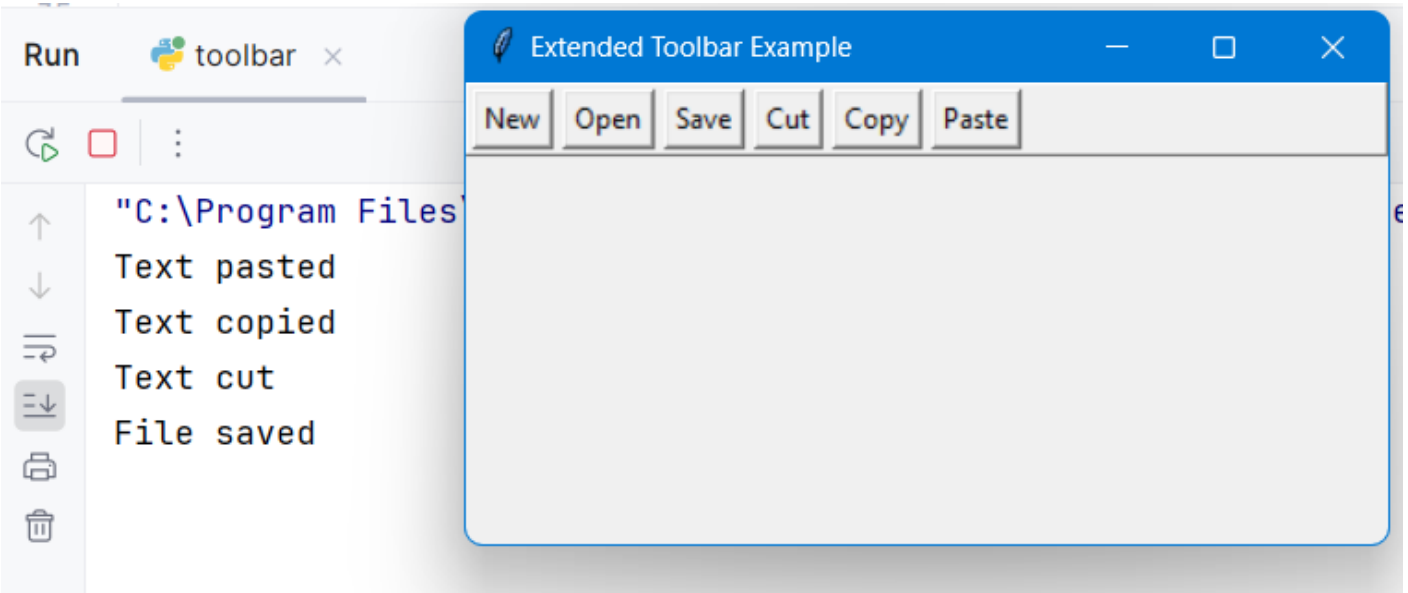
Explanation - Example of Toolbar

- Root Window: The main application window is created using `tk.Tk()`.
- Toolbar Frame:
  - Created using `tk.Frame(root, bd=1, relief=tk.RAISED)`.
  - Configured with a border (bd) and a raised relief style (relief).
  - Placed at the top of the root window using `toolbar.pack(side=tk.TOP, fill=tk.X)`.
- Buttons:
  - Created using `tk.Button(toolbar, text="Button Text", command=command_function)`.
  - Added to the toolbar frame using `button.pack(side=tk.LEFT, padx=2, pady=2)`.
- Button Commands: Each button is linked to a function that prints a message to the console.

Toolbar

TRY IT

Extend the previous example by adding a "Cut", "Copy", and "Paste" button to the toolbar. Each button should print a corresponding message to the console when clicked.



6. Scale Widget

ICTACADEMY

Scale widget

The scale widget in Tkinter is a graphical slider that allows users to select a value within a specified numeric range. It provides a way to interactively adjust and input numeric values using a GUI interface.

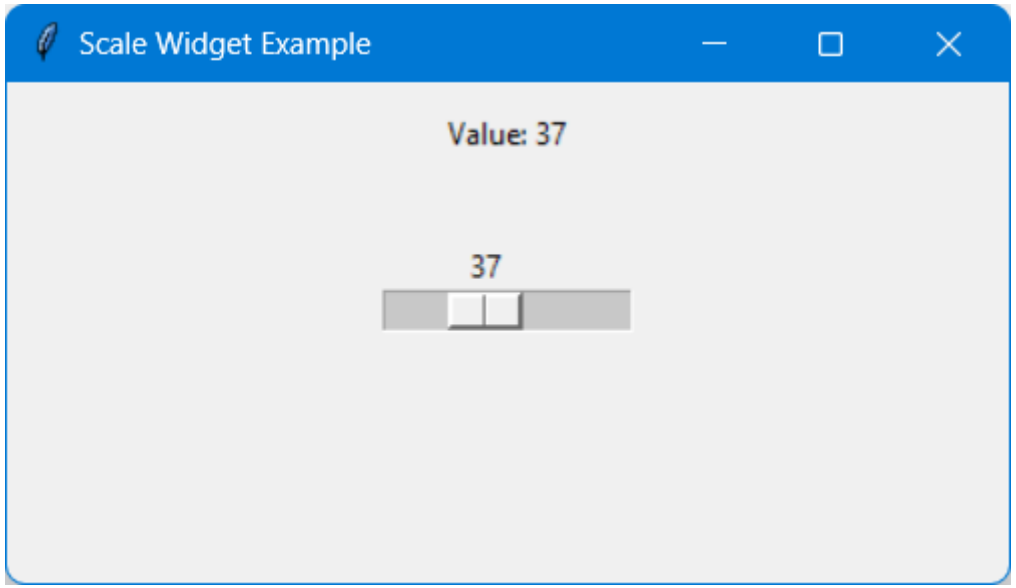
Steps to create a scale widget:

1. Create the Root Window: Initialize the main application window.
2. Create a Frame for the Toolbar: This frame will hold the scale widget.
3. Add the Scale Widget:
   - Create the scale widget (tk.Scale) within the frame.
   - Specify the parent widget (frame), the range of values (from_ and to parameters), orientation (orient parameter), and a function (command parameter) to handle value changes.
4. Pack the Scale Widget

Scale widget

Example – GUI of Scale widget

Create a Tkinter application with a scale widget that scales from 0 to 100.



Scale widget

Example – Code of Scale widget

scaleWidget.py

```python
import tkinter as tk

# Function to handle scale value changes
1 usage
def on_scale_change(value):
    label_value.config(text=f"Value: {value}")

# Create the main application window
root = tk.Tk()
root.title("Scale Widget Example")
root.geometry("400x200")
# Create a frame for the scale widget
frame = tk.Frame(root)

```

```python
# Create a scale widget
scale = tk.Scale(frame, from_=0, to=100, orient=tk.HORIZONTAL, command=on_scale_change)
scale.pack()

# Label to display scale value
label_value = tk.Label(root, text="Value: 0")
label_value.pack(pady=10)

# Pack the frame containing the scale widget
frame.pack(padx=20, pady=20)

# Start the Tkinter event loop
root.mainloop()
```

ojectAlgoritmo > scaleWidget.py

Scale widget

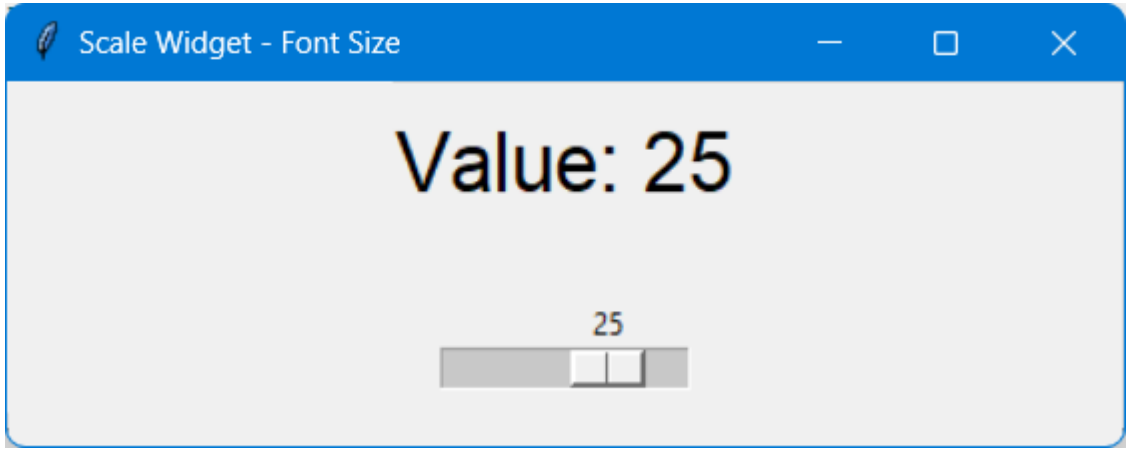
Explanation - Example of Scale widget

- `scale = tk.Scale(…)`: Creates a horizontal scale widget (tk.Scale) within the frame.
- Parameters:
  - `from_=0 and to=100`: Specifies the range of values for the scale widget (from 0 to 100).
  - `orient=tk.HORIZONTAL`: Specifies the orientation of the scale widget (horizontal).
  - `command=on_scale_change`: Specifies the function (on_scale_change) to call during scale value changes.

Scale widget

TRY IT

Enhance the scale widget example by adding a label that changes its font size based on the selected scale value.



7. SpinBox Widget

ICTACADEMY

SpinBox widget

The SpinBox widget in Tkinter is a graphical control element consisting of a text box and two small arrow buttons that allow users to select a numeric value within a specified range by incrementing or decrementing.

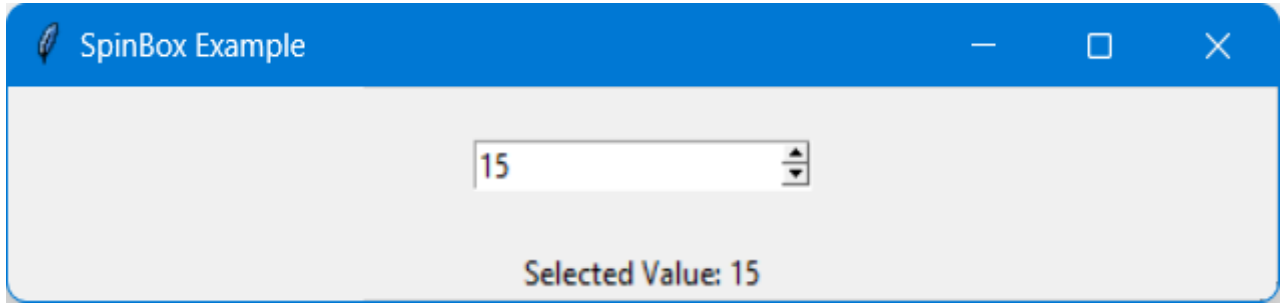
Steps to create a SpinBox widget:

1. Create the Main Application Window: root = tk.Tk()
2. Create SpinBox Widget: spinbox = tk.Spinbox(root, from_=0, to=100)
3. Pack SpinBox Widget: spinbox.pack()
4. Handle SpinBox Value Changes: Define a function on_spinbox_change() to update based on SpinBox value.

SpinBox widget

Example – GUI of SpinBox widget

Create a Tkinter application with a spinbox widget that spins from 0 to 100.



SpinBox widget

Example – Code of SpinBox widget

spinboxWidget.py

```python
import tkinter as tk

1 usage
def on_spinbox_change():
    selected_value = spinbox.get()
    label_value.config(text=f"Selected Value: {selected_value}")

# Create the main application window
root = tk.Tk()
root.title("SpinBox Example")

# Create SpinBox widget
spinbox = tk.Spinbox(root, from_=0, to=100, command=on_spinbox_change)
spinbox.pack(pady=20)

# Label to display selected value
label_value = tk.Label(root, text="Selected Value: 0")
label_value.pack()

# Start the Tkinter event loop
root.mainloop()
```

SpinBox widget

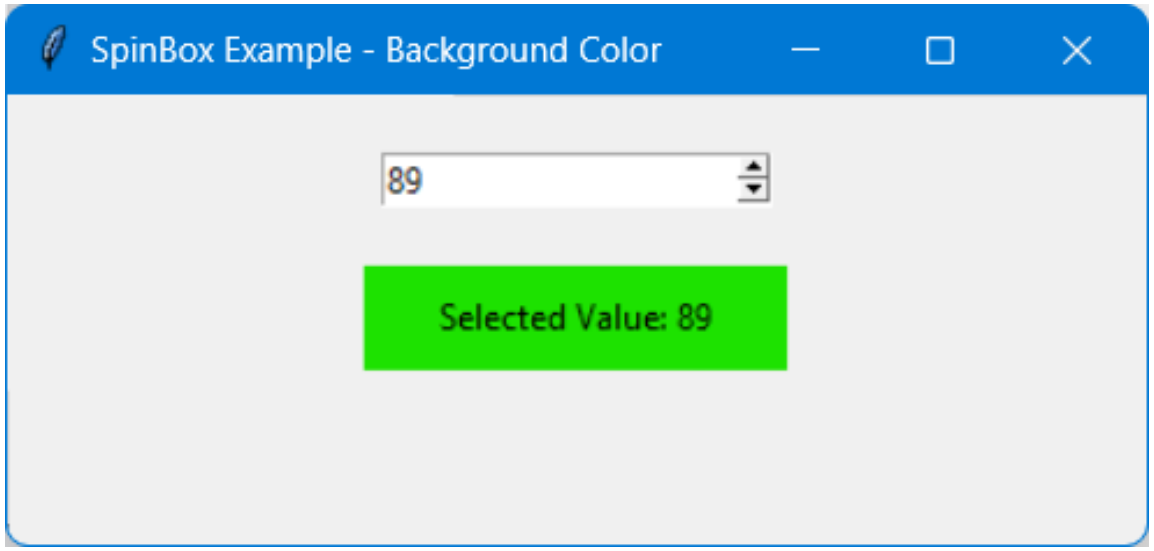
Explanation - Example of SpinBox widget

- Defines a function on_spinbox_change that:
  - Retrieves the current value of the SpinBox using spinbox.get().Updates the text of label_value to display the selected value.
- spinbox = tk.Spinbox(root, …)
  - Creates a SpinBox widget “spinbox” within the main window “root”.
  - Parameters:
    - from_=0 and to=100: Specifies the range of values (from 0 to 100).
    - command=on_spinbox_change: Specifies the function (on_spinbox_change) to call whenever the SpinBox value changes.

SpinBox widget

TRY IT

Enhance the SpinBox example to dynamically change the background color of a label based on the selected value of the SpinBox.



8. MessageBox

MessageBox

MessageBox

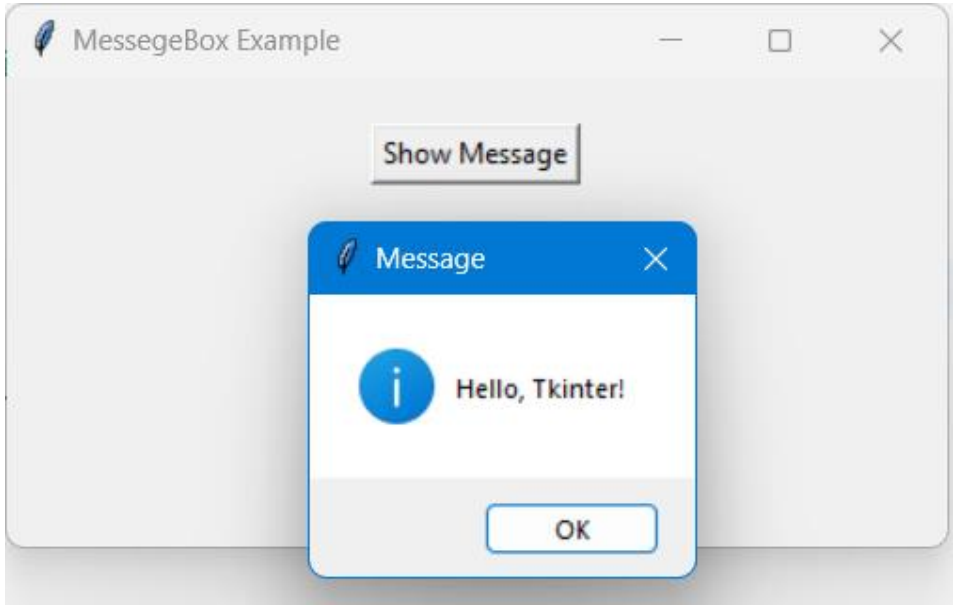
Tkinter Messagebox provides dialog boxes to show messages to users, including alerts, warnings, and confirmation prompts.

Step to create a messegebox widget:

```
import tkinter.messagebox as mb
```

Example – GUI of MessageBox

Create a Tkinter application that display a prompt to user upon a button click.



MessageBox

Example – Code of MessageBox

messegeBox.py

```python
import tkinter as tk
import tkinter.messagebox as mb

1 usage
def show_message():
    mb.showinfo( title: "Message",  message: "Hello, Tkinter!")

root = tk.Tk()
root.title("MessegeBox Example")
root.geometry("400x200")

tk.Button(root, text="Show Message", command=show_message).pack(pady=20)
root.mainloop()
```

MessageBox

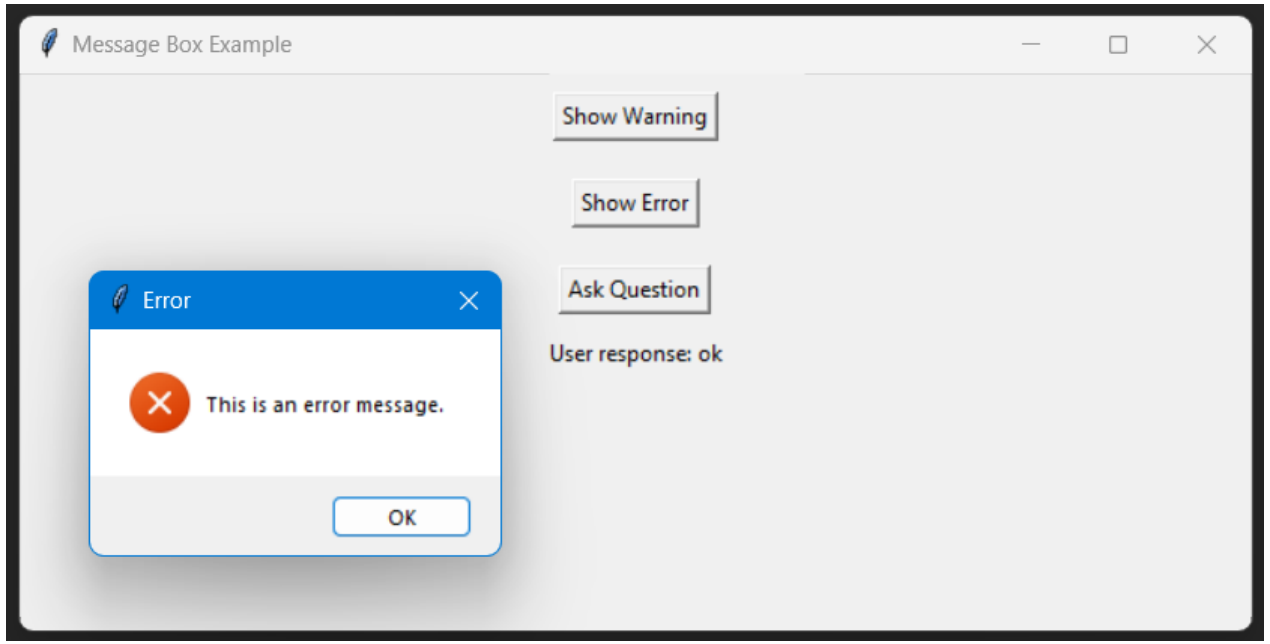
Explanation - Example of MessageBox

- Imports:
  - `import tkinter.messagebox as mb` – it Imports the Tkinter messagebox module as mb for convenience.
- Function `show_message()`:
  - Defines a function `show_message` that displays an informational message box using `mb.showinfo`.

MessageBox

TRY IT

Enhance the example to include different types of message boxes (warning, error, question) and handle user responses (e.g., display a different message or take different actions based on user input).



9. Graphics and Shapes - Line Graphics

Graphics and shapes - line graphics

Line

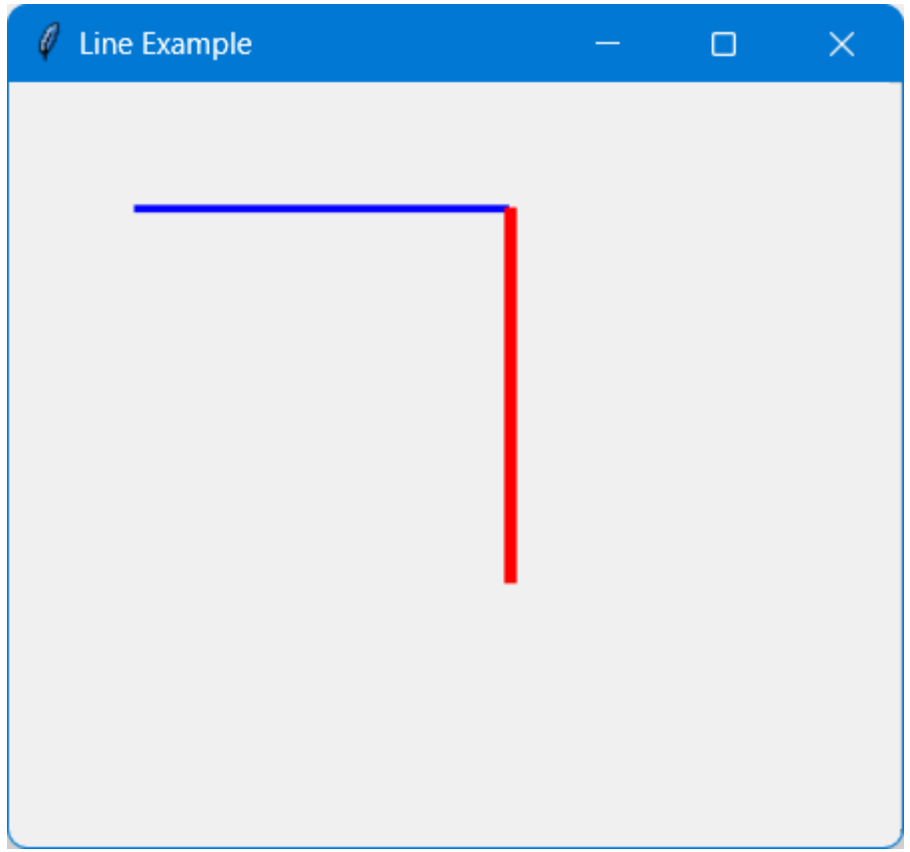
Tkinter allows drawing lines between specified points using the Canvas widget.

Step to create a line:

```
canvas.create_line(x1, y1, x2, y2, ...)
```

Example – GUI of Line

Create a Tkinter application that display a line on canvas.



Graphics and shapes - line graphics

Example – Code of Line

line.py

```python
import tkinter as tk

root = tk.Tk()
root.title("Line Example")
root.geometry("500x400")

canvas = tk.Canvas(root, width=400, height=400)
canvas.pack()

canvas.create_line(50, 50, 200, 50, fill="blue", width=3)
canvas.create_line(200, 50, 200, 200, fill="red", width=5)

root.mainloop()
```

Graphics and shapes - line graphics

Explanation - Example of Line

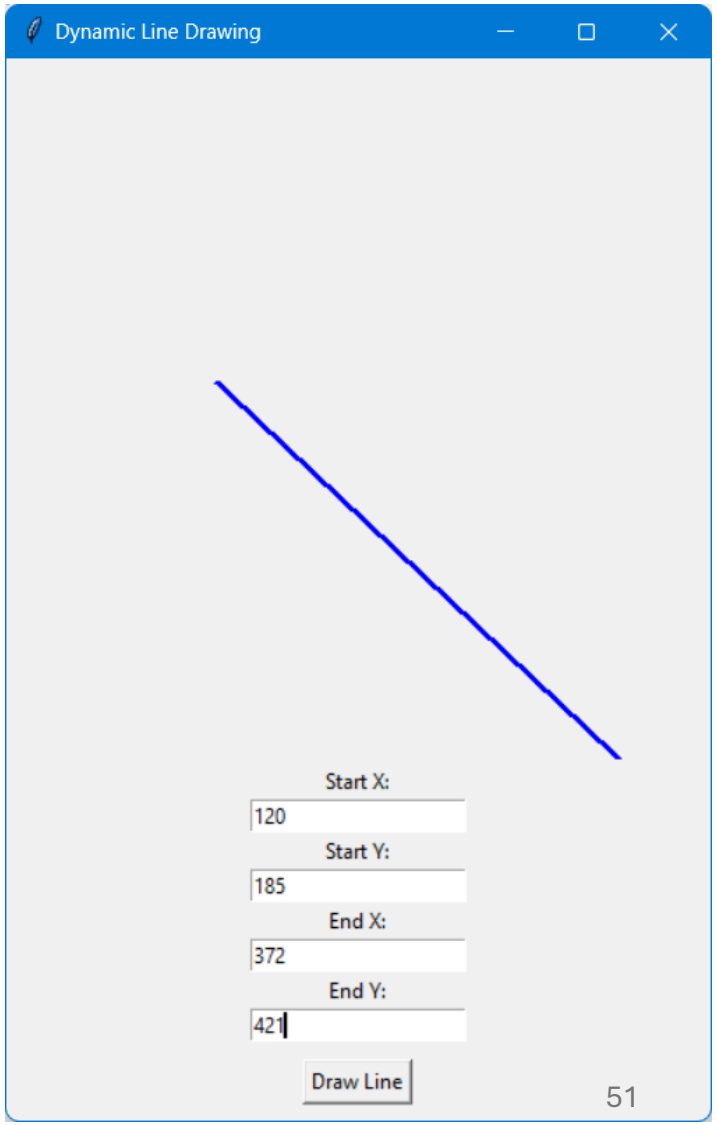
- Canvas Widget:
  - Creates a Canvas widget (canvas) within the main window (root) with specified width and height.

- `create_line` Method:
  - Uses `canvas.create_line` to draw lines:
    - `canvas.create_line(50, 50, 200, 50, fill="blue", width=3)`: Draws a blue line from (50, 50) to (200, 50) with a width of 3 pixels.
    - `canvas.create_line(200, 50, 200, 200, fill="red", width=5)`: Draws a red line from (200, 50) to (200, 200) with a width of 5 pixels.

Graphics and shapes - line graphics

TRY IT

Extend the example by allowing users to input coordinates via entry widgets and draw lines dynamically based on those coordinates.



10. Graphics and Shapes - Box Graphics

Graphics and shapes - box graphics

Box

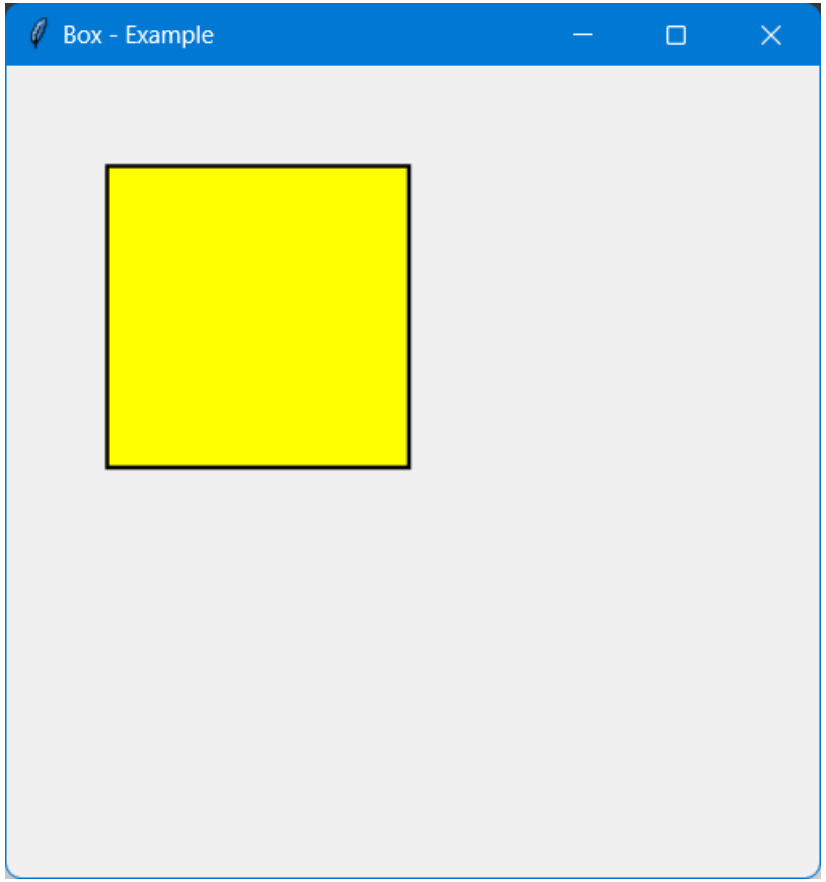
Tkinter Canvas widget allows drawing rectangles or boxes with specified coordinates and attributes.

Step to create a box:

```
canvas.create_rectangle(x1, y1, x2, y2, ...)
```

Example – GUI of Box

Create a Tkinter application that display a box on canvas.



Graphics and shapes - box graphics

Example – Code of Box

box.py

```python
import tkinter as tk

root = tk.Tk()
root.title("Box - Example")

canvas = tk.Canvas(root, width=400, height=400)
canvas.pack()

canvas.create_rectangle(50, 50, 200, 200, outline="black", fill="yellow", width=2)

root.mainloop()

```

Graphics and shapes - box graphics

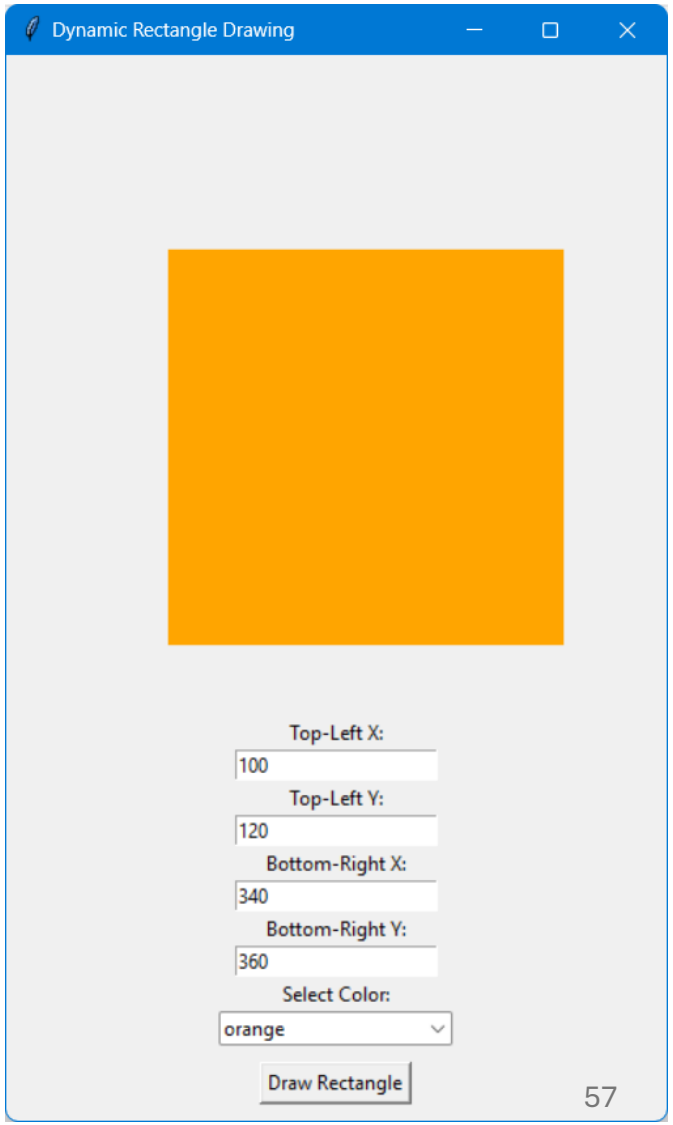
Explanation - Example of Box

- `create_rectangle` Method:
  - Uses `canvas.create_rectangle` to draw a rectangle:
    - `canvas.create_rectangle(50, 50, 200, 200, outline="black", fill="yellow", width=2)`: Draws a rectangle with top-left corner at (50, 50) and bottom-right corner at (200, 200).
    - `outline="black"`: Sets the outline color to black.
    - `fill="yellow"`: Fills the rectangle with yellow color.
    - `width=2`: Sets the outline width to 2 pixels.

Graphics and shapes - box graphics

TRY IT

Enhance the example to allow users to change the size, color, and position of the rectangle using entry widgets or dropdown menus.



11. Graphics and Shapes - Canvas

Graphics and shapes - canvas

Canvas

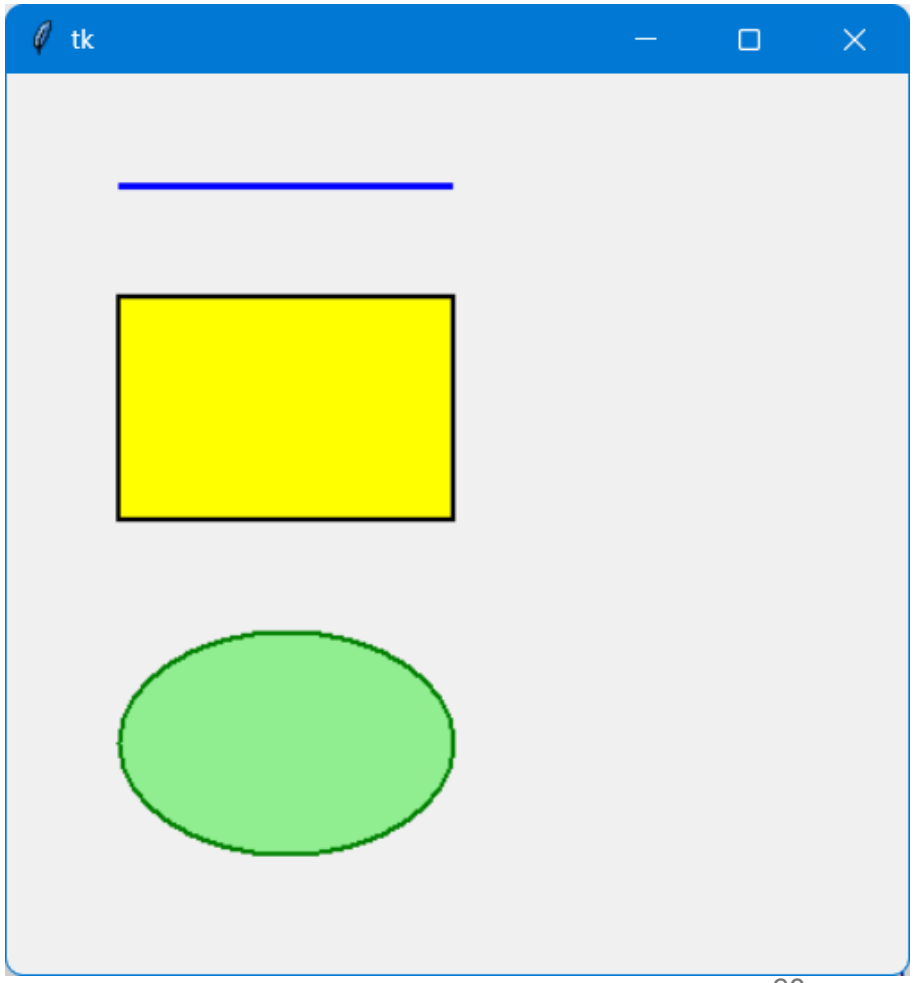
Tkinter Canvas widget provides a drawing space to create graphics, shapes, and images.

Step to create a canvas:

```
canvas = tk.Canvas(root, width=400, height=400)
```

Example – GUI of Canvas

Create a Tkinter application that makes a canvas placing some shapes on it.



Graphics and shapes - canvas

Example – Code of Canvas

canvas.py

```python
import tkinter as tk

root = tk.Tk()
canvas = tk.Canvas(root, width=400, height=400)
canvas.pack()

canvas.create_line(50, 50, 200, 50, fill="blue", width=3)
canvas.create_rectangle(50, 100, 200, 200, outline="black", fill="yellow", width=2)
canvas.create_oval(50, 250, 200, 350, outline="green", fill="lightgreen", width=2)

root.mainloop()
```

Graphics and shapes - canvas

Explanation - Example of Canvas

- Canvas Widget:
  - Creates a Canvas widget (canvas) within the main window (root) with specified width and height.
- Drawing Methods:
  - Uses `canvas.create_line`, `canvas.create_rectangle`, and `canvas.create_oval` methods to draw:
    - A blue line from (50, 50) to (200, 50) with a width of 3 pixels.
    - A yellow rectangle with outline in black, from (50, 100) to (200, 200) with a width of 2 pixels.
    - A light green oval with outline in green, from (50, 250) to (200, 350) with a width of 2 pixels.

12. Graphics and Shapes - Images in GUI

Graphics and shapes – images in GUI

Images in GUI

Tkinter allows showing images (e.g., PNG, JPEG) within GUI applications using the PhotoImage class.

Step to add an image:

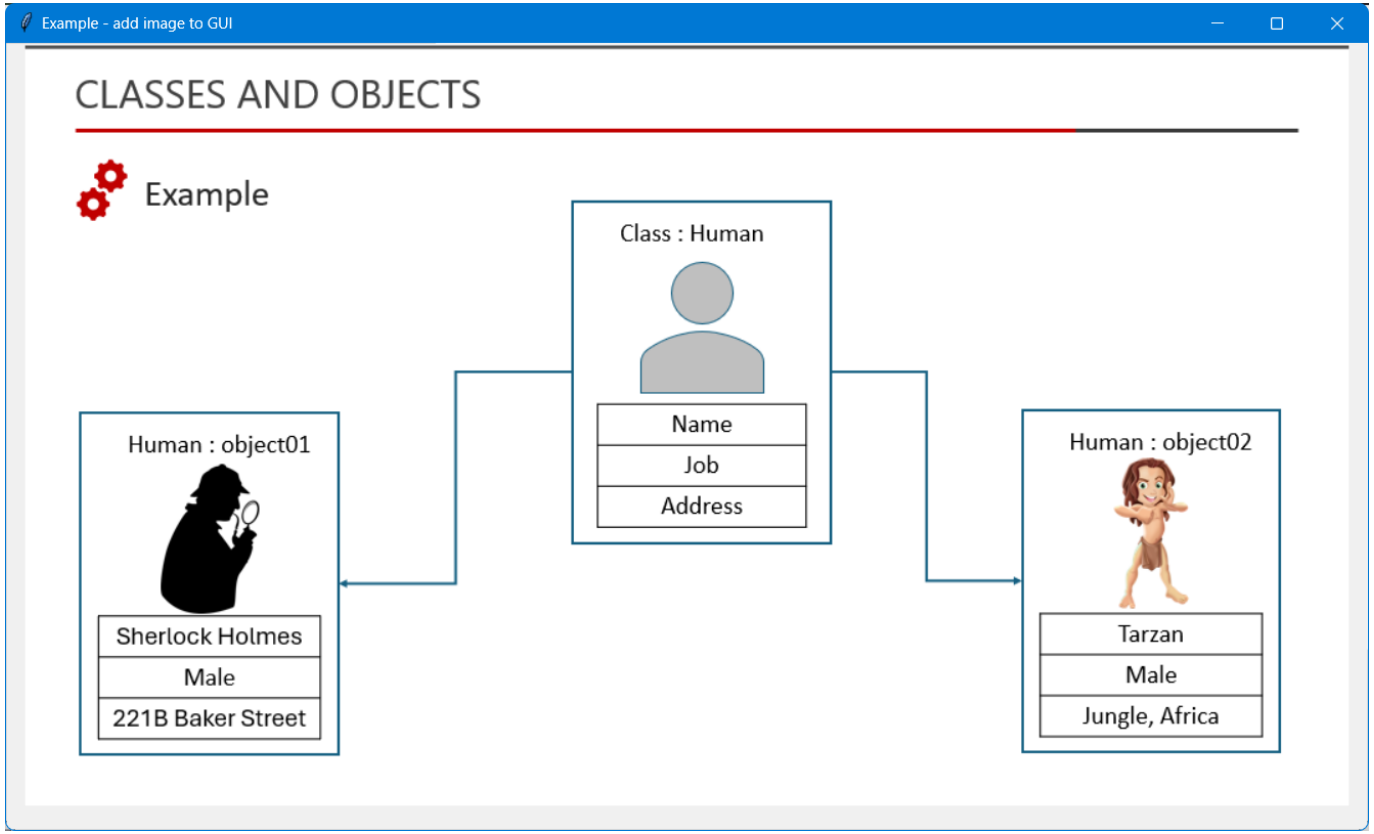
```
photo = tk.PhotoImage(file="image.png")
```

Graphics and shapes – images in GUI

In Tkinter some image formats are not supported by default. How can we address that issue?

Example – showing an image

Create a Tkinter application that shows an image.



Graphics and shapes – images in GUI

Example – Code of showing an image

addImage.py

```python
import tkinter as tk

root = tk.Tk()
root.title("Example - add image to GUI")

# Load and display an image
photo = tk.PhotoImage(file="D:/Algoritmo/test.png")
label = tk.Label(root, image=photo)
label.pack()

root.mainloop()

```

Explanation - Example of Canvas

- PhotoImage Class:

Uses `tk.PhotoImage(file="image.png")` to create a PhotoImage object (photo) from an image file (image.png).

Question?

Thank you