Tkinter Tutorial: Building GUI Applications in Python

---

1. Introduction to Tkinter

- **Tkinter**: A built-in Python library for creating GUI (Graphical User Interface) applications.
- **Features**:
  - Easy to use.
  - Cross-platform.
  - Comes pre-installed with Python.

---

2. Creating Your First Tkinter Window

Basic Tkinter Window

Here's how you create a simple window:

```
import tkinter as tk

# Create the main window
root = tk.Tk()

# Set window title
root.title("My First Tkinter Window")

# Set window size
root.geometry("400x300")

# Run the application
root.mainloop()
```

Key Points:

- `tk.Tk()`: Initializes the main application window.
- `root.mainloop()`: Starts the event loop for the GUI.

Adding a Label to the Window

```
import tkinter as tk

root = tk.Tk()
```

```
root.title("Label Example")

# Create a label
label = tk.Label(root, text="Hello, Tkinter!", font=("Arial", 16))

# Place the label
label.pack()

root.mainloop()
```

Explanation:

- `tk.Label()`: Creates a text label.
- `.pack()`: Places the widget in the window.

---

3. Common Tkinter Widgets

3.1 Button

```
import tkinter as tk

def on_click():
    print("Button clicked!")

root = tk.Tk()
root.title("Button Example")

# Create a button
button = tk.Button(root, text="Click Me", command=on_click)

# Place the button
button.pack()

root.mainloop()
```

Key Points:

- `command=on_click`: Executes a function when the button is clicked.

3.2 Entry (Text Input)

```
import tkinter as tk

def show_input():
    print(entry.get())

root = tk.Tk()
root.title("Entry Example")

# Create an entry widget
entry = tk.Entry(root, width=20)
```

```
# Create a button to display the input
button = tk.Button(root, text="Show Input", command=show_input)

entry.pack()
button.pack()

root.mainloop()
```

3.3 Checkbutton (Checkbox)

```
import tkinter as tk

def display_choice():
    print(var.get())

root = tk.Tk()
root.title("Checkbutton Example")

# Create a Tkinter variable
var = tk.BooleanVar()

# Create a checkbox
checkbox = tk.Checkbutton(root, text="Check Me", variable=var,
command=display_choice)

checkbox.pack()

root.mainloop()
```

3.4 Radiobutton

```
import tkinter as tk

def display_selection():
    print(f"Selected: {var.get()}")

root = tk.Tk()
root.title("Radiobutton Example")

# Create a Tkinter variable
var = tk.StringVar(value="Option 1")

# Create radiobuttons
radio1 = tk.Radiobutton(root, text="Option 1", variable=var, value="Option
1", command=display_selection)
radio2 = tk.Radiobutton(root, text="Option 2", variable=var, value="Option
2", command=display_selection)

radio1.pack()
radio2.pack()

root.mainloop()
```

3.5 Listbox

```
import tkinter as tk

def display_selection(event):
    selected_item = listbox.get(listbox.curselection())
    print(f"Selected: {selected_item}")

root = tk.Tk()
root.title("Listbox Example")

# Create a listbox
listbox = tk.Listbox(root)
listbox.insert(1, "Python")
listbox.insert(2, "Java")
listbox.insert(3, "C++")

# Bind selection event
listbox.bind("<<ListboxSelect>>", display_selection)

listbox.pack()

root.mainloop()
```

---

4. Layout Management

4.1 pack()

- Places widgets in a linear fashion (top to bottom or left to right).

Example:

```
label1 = tk.Label(root, text="Top")
label2 = tk.Label(root, text="Bottom")

label1.pack(side="top")
label2.pack(side="bottom")
```

4.2 grid()

- Aligns widgets in a grid layout.

Example:

```
label1 = tk.Label(root, text="Row 0, Col 0")
label2 = tk.Label(root, text="Row 1, Col 1")

label1.grid(row=0, column=0)
label2.grid(row=1, column=1)
```

4.3 place()

- Places widgets at specific coordinates.

Example:

```
label = tk.Label(root, text="Placed Label")
label.place(x=100, y=50)
```

---

5. Event Handling

- Handle user interactions like clicks, key presses, etc.

Example: Binding Key Events

```
def on_key_press(event):
    print(f"Key pressed: {event.char}")

root.bind("<Key>", on_key_press)
```

---

6. Creating a Simple App: Calculator

```
import tkinter as tk

def calculate():
    result = eval(entry.get())
    label_result.config(text=f"Result: {result}")

root = tk.Tk()
root.title("Calculator")

entry = tk.Entry(root, width=20)
button = tk.Button(root, text="Calculate", command=calculate)
label_result = tk.Label(root, text="Result: ")

entry.pack()
button.pack()
label_result.pack()

root.mainloop()
```

---

7. Advanced Features

7.1 Adding Menus

```
def on_new():
    print("New file created")

root = tk.Tk()
root.title("Menu Example")

menu = tk.Menu(root)
root.config(menu=menu)

file_menu = tk.Menu(menu)
menu.add_cascade(label="File", menu=file_menu)
```

```
file_menu.add_command(label="New", command=on_new)
file_menu.add_command(label="Exit", command=root.quit)

root.mainloop()
```

7.2 Canvas for Drawing

```
def draw_circle(event):
    canvas.create_oval(event.x-10, event.y-10, event.x+10, event.y+10,
fill="blue")

root = tk.Tk()
root.title("Canvas Example")

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

canvas.bind("<Button-1>", draw_circle)

root.mainloop()
```

---

8. Building Real-World Applications

Ideas:

1. **To-Do List**: Use `Listbox` and `Entry`.
2. **Weather App**: Fetch data using an API and display it in labels.
3. **Game**: Build a simple game using `Canvas`.

---

9. Conclusion

- Tkinter is a powerful tool for creating GUIs in Python.
- Practice is key! Try building your projects to strengthen your skills.
