User Guide: React-like Tkinter App in Python

This guide explains a React-like application built with Python's tkinter module. It mimics React's useState and useEffect features for managing state and side effects.

Module 1: Imports and App Initialization

```
import tkinter as tk


class ReactLikeApp:
    def __init__(self, root):
        self.root = root
        self.root.title("React-like Tkinter App")
```

Module 2: State and Effect Management

```
        # useState equivalent
        self.counter = tk.IntVar(value=0)

        # useEffect equivalent: run a function when counter changes
        self.counter.trace_add("write", self.on_counter_change)
```

Module 3: UI Setup

```
        # UI setup
        self.label = tk.Label(root, text="Counter: 0", font=("Arial", 18))
        self.label.pack(pady=20)

        self.button = tk.Button(root, text="Increment", command=self.increment)
        self.button.pack(pady=10)

        self.reset_button = tk.Button(root, text="Reset", command=lambda: self.counter.set(0))
```

```
        self.reset_button.pack(pady=5)
```

Module 4: State Update and Effect Function

```
    def increment(self):
        # setState equivalent
        self.counter.set(self.counter.get() + 1)


    def on_counter_change(self, *args):
        # useEffect equivalent - auto-updates when counter changes
        value = self.counter.get()
        self.label.config(text=f"Counter: {value}")
        print(f"[Effect] Counter changed to {value}")
```

Module 5: Application Entry Point

```
if __name__ == "__main__":
    root = tk.Tk()
    app = ReactLikeApp(root)
    root.mainloop()
```