Worksheet: Understanding a Stopwatch Application with Tkinter

Instructions:

Study the provided code for a stopwatch application built using `tkinter`. After reading through the code, answer the questions to deepen your understanding of how the program works. You can run the code and experiment with it to test different ideas.

Code: Stopwatch Application (Without Class)

```
import tkinter as tk

# Initialize main window
root = tk.Tk()
root.title("Stopwatch")

# Variables to track time and running state
time_elapsed = 0
is_running = False

# Function to update the time
def update_time():
    if is_running:
        global time_elapsed
        time_elapsed += 1
        hours = time_elapsed // 3600
        minutes = (time_elapsed // 60) % 60
        seconds = time_elapsed % 60
        time_label.config(text=f"{hours:02}:{minutes:02}:{seconds:02}")
        root.after(1000, update_time)

# Function to start the stopwatch
def start():
    global is_running
    if not is_running:
        is_running = True
        update_time()

# Function to stop the stopwatch
def stop():
    global is_running
    if is_running:
        is_running = False

# Function to reset the stopwatch
def reset():
    global time_elapsed, is_running
    is_running = False
    time_elapsed = 0
    time_label.config(text="00:00:00")

# Create label to display time
```

```
time_label = tk.Label(root, text="00:00:00", font=("Arial", 40))
time_label.grid(row=0, column=0, columnspan=3, pady=20)

# Create buttons for start, stop, and reset
start_button = tk.Button(root, text="Start", command=start, width=10)
start_button.grid(row=1, column=0, padx=5, pady=5)

stop_button = tk.Button(root, text="Stop", command=stop, width=10)
stop_button.grid(row=1, column=1, padx=5, pady=5)

reset_button = tk.Button(root, text="Reset", command=reset, width=10)
reset_button.grid(row=1, column=2, padx=5, pady=5)

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

Questions:

1. **Initialization**
   - What does the following line do?

     ```
     root = tk.Tk()
     ```

   - **Hint**: Think about how the tkinter library manages the main window.
2. **Tracking Time and State**
   - What are `time_elapsed` and `is_running` used for in this program?
   - Why do we declare them as `global` inside the functions `start()`, `stop()`, and `reset()`?
3. **Event Handling: Start Button**
   - When the "Start" button is clicked, what happens in the `start()` function?
   - How does the `start()` function ensure that multiple clicks don't restart the timer when it's already running?
4. **Updating the Stopwatch Display**
   - What is the purpose of the `update_time()` function?
   - How does the function know when to update the time? Which method makes sure it is called repeatedly?
5. **Time Formatting**
   - In `update_time()`, time is converted to hours, minutes, and seconds. What does this line do?

     ```
     hours = time_elapsed // 3600
     minutes = (time_elapsed // 60) % 60
     seconds = time_elapsed % 60
     ```

   - Why do we need to use `//` and `%` operators here?
6. **Stopping the Stopwatch**

    - What happens when the "Stop" button is clicked? Describe the behavior of the `stop()` function.
    - Why is it important to have the `is_running` check in the `stop()` function?
7. **Resetting the Stopwatch**
    - How does the `reset()` function reset the stopwatch to `00:00:00`?
    - Besides resetting the time, what else does the `reset()` function do to ensure the stopwatch stops?
8. **Button Events and Layout**
    - Look at how the buttons are created:

      ```
      start_button = tk.Button(root, text="Start", command=start, width=10)
      ```

        - What does the `command=start` part do?
        - How is the `grid()` method being used here to position the buttons and labels on the window?
9. **Understanding the Main Loop**
    - Why is `root.mainloop()` necessary at the end of the program?
    - What would happen if we removed this line?

Extended Challenges (Optional):

1. **Add Milliseconds**: Modify the program so that it displays milliseconds (1/1000 of a second).
2. **Pause and Resume**: Add a new "Pause" button to the stopwatch that can pause the time and resume it without resetting the elapsed time.
3. **Laps Feature**: Create a "Lap" button that stores and displays the current time when clicked, allowing users to track laps.

Submission Instructions:

- Answer the questions based on your understanding of the code.
- For the extended challenges, feel free to modify the code and include the updated version with your answers.