TEACHNOOK INTERNSHIP MINOR PROJECT

—Create A Countdown Timer Using Python

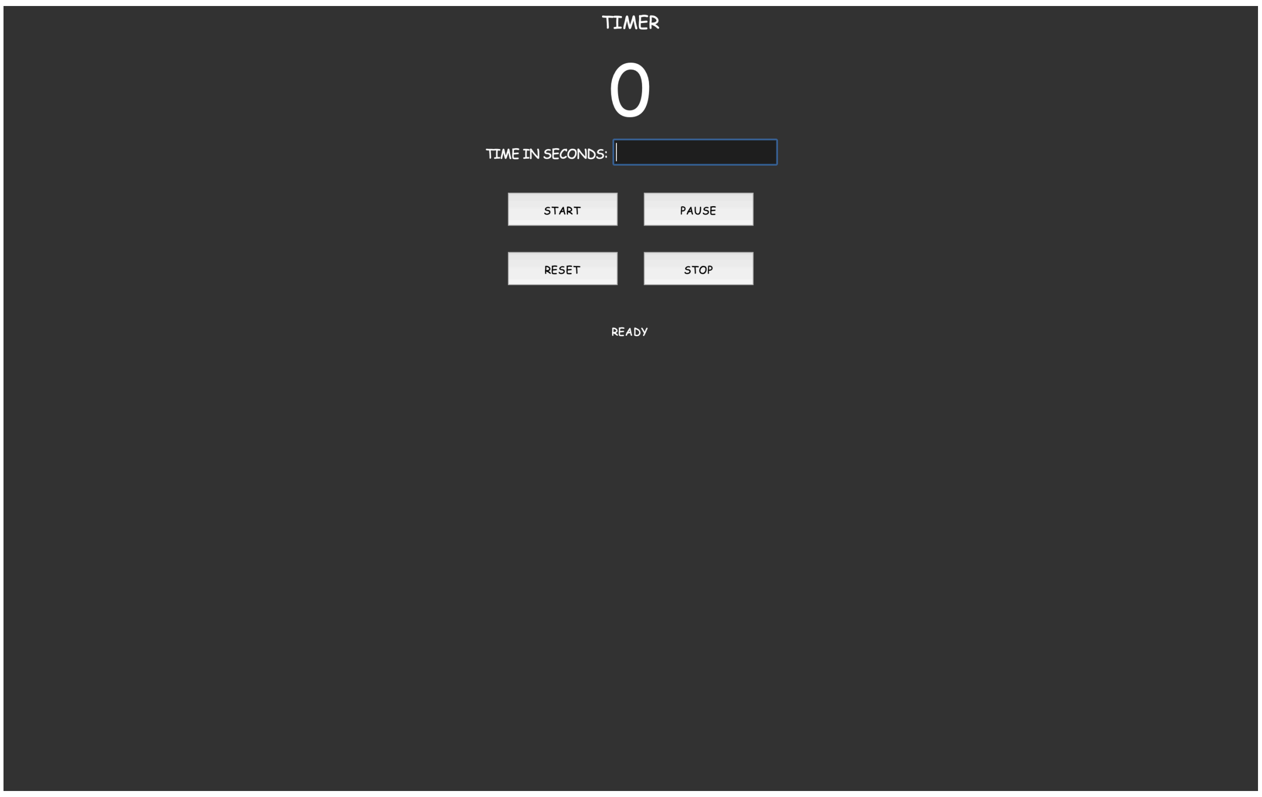
Features To Include

Reset/ Stop Pause /Resume

Here’s the solution code for the given problem above:

```
from tkinter import *
import time
from threading import Thread


class Timer:
    def __init__(self, label, status_label):
        self.label = label
        self.limit = None
        self.isRunning = False
        self.status_label = status_label
        self.isReset = False
        self.isStopped = False
        self.isPaused = False

    def start(self):
        t = Thread(target=self.__start)
        t.start()
```

```python
    def __start(self):
        self.status_label.config(fg="white")
        if not self.limit or self.isRunning:
            return
        self.isRunning = True
        while self.limit >= 0 if self.limit else self.isRunning:
            if (not self.isRunning):
                break
            self.label.config(text=self.limit)
            self.limit -= 1
            time.sleep(1)
        self.isRunning = False
        fg = "white"
        status = "Time's up!"
        if self.isPaused:
            status = "Paused"
        elif self.isStopped:
            status = "Stopped"
        elif self.isReset:
            status = "Reset"
        else:
            fg = "white"
        self.status_label.config(text=status,fg="white")
        self.status_label.config(fg=fg)
        if not self.isPaused:
            self.limit = None
        self.isReset = False
        self.isStopped = False
        self.isPaused = False

    def pause(self):
        self.isRunning = False


def main():
    # GUI window
    root = Tk()
    root.title("TIMER")
    root.geometry('500x500')
    label = Label(root, text="TIMER", font=("Comic Sans MS",
20))
    label.pack()
    countdown = Label(root, text="0", font=("Comic Sans MS",
80))
    countdown.pack()
    entry_frame = Frame(root)
    enter_number_label = Label(entry_frame, text="TIME IN
SECONDS:", font=("Comic Sans MS", 15), padx=2, pady=2)
    enter_number_label.pack(side=LEFT)
    enter_number = Entry(entry_frame, font=("Comic Sans MS",
15))
    enter_number.pack(side=RIGHT)
    enter_number.focus()
    entry_frame.pack()

    button_frame = Frame(root, padx=15, pady=15)

    button_col_1_frame = Frame(button_frame, relief='raised')

    button_col_2_frame = Frame(button_frame, relief='raised')

    start_button = Button(button_col_1_frame, text="START",
font=("Comic Sans MS", 12), width=12, height=2,
relief='raised',

                          bg='white', fg='black')

    start_button.pack(side=LEFT, anchor=CENTER, pady=12,
padx=12)

    pause_button = Button(button_col_1_frame, text="PAUSE",
font=("Comic Sans MS", 12), width=12, height=2,
relief='raised',

                          bg='white', fg='black')

    pause_button.pack(side=LEFT, anchor=CENTER, pady=12,
padx=12)

    reset_button = Button(button_col_2_frame, text="RESET",
font=("Comic Sans MS", 12), width=12, height=2,
relief='raised',

                          bg='white', fg='black')

    reset_button.pack(side=LEFT, anchor=CENTER, pady=12,
padx=12)

    stop_button = Button(button_col_2_frame, text="STOP",
font=("Comic Sans MS", 12), width=12, height=2,
relief='raised',

                         bg='white', fg='black')

    stop_button.pack(side=LEFT, anchor=CENTER, pady=12,
padx=12)

    button_col_1_frame.pack(fill=X)

    button_col_2_frame.pack(fill=X)
    button_frame.pack()
    status_label = Label(root, text="READY", font=("Comic 
Sans MS", 12), padx=12, pady=12)
    status_label.pack()
    t = Timer(countdown, status_label)

    def start_timer():
        if not t.limit:
            try:
                t.limit = int(enter_number.get())
            except ValueError:
                status_label.config(fg="white")
                status_label.config(text="Enter a valid 
number")
                time.sleep(1)
                status_label.config(text="Ready")
                status_label.config(fg="white")
        t.start()
        status_label.config(text="Running", fg="white")


    def pause_timer():
        if not t.limit:
            return
        if t.isRunning:
            t.isPaused = True
            t.pause()
            pause_button.config(text="RESUME")

            status_label.config(text="Paused")
            status_label.config(fg="white")
        else:
            pause_button.config(text="Pause")
            t.isPaused = False
            t.start()
            status_label.config(text="Running", fg="white")
            status_label.config(fg="white")

    def reset_timer():
        t.isReset = True
        t.isRunning = False
        countdown.config(text="0")
        pause_button.config(text="Pause")

        status_label.config(text="Reset")
        status_label.config(fg="white")

    def stop_timer():
        t.isStopped = True
        t.isRunning = False
        pause_button.config(text="Pause")

        status_label.config(text="Stopped")
        status_label.config(fg="white")
    start_button.config(command=start_timer)
    pause_button.config(command=pause_timer)
    stop_button.config(command=stop_timer)
    reset_button.config(command=reset_timer)

    root.mainloop()


if __name__ == '__main__':
    main()
```


TIMER
0
TIME IN SECONDS:
START
PAUSE
RESET
STOP
READY