Countdown Timer

using Python

DS-MINOR-JULY

The idea

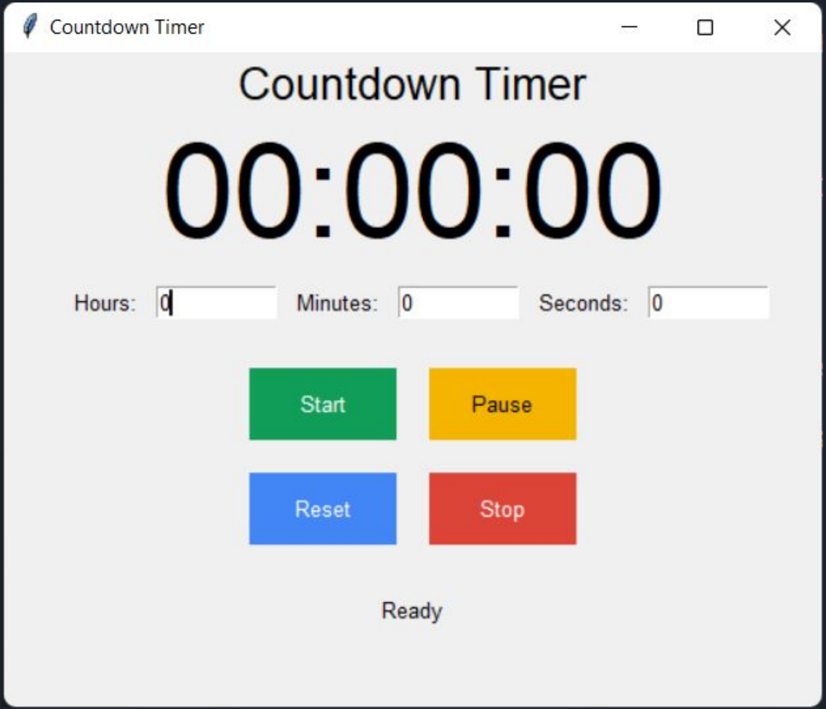
The idea is to make a countdown timer in python so that it enables users to create a timer which counts down to 0 when started.
It should also contain capabilities to Pause, Resume, Stop and Reset the timer.
The GUI based solution of the program will be more handy and helpful for user.

Specifications:

The program uses:

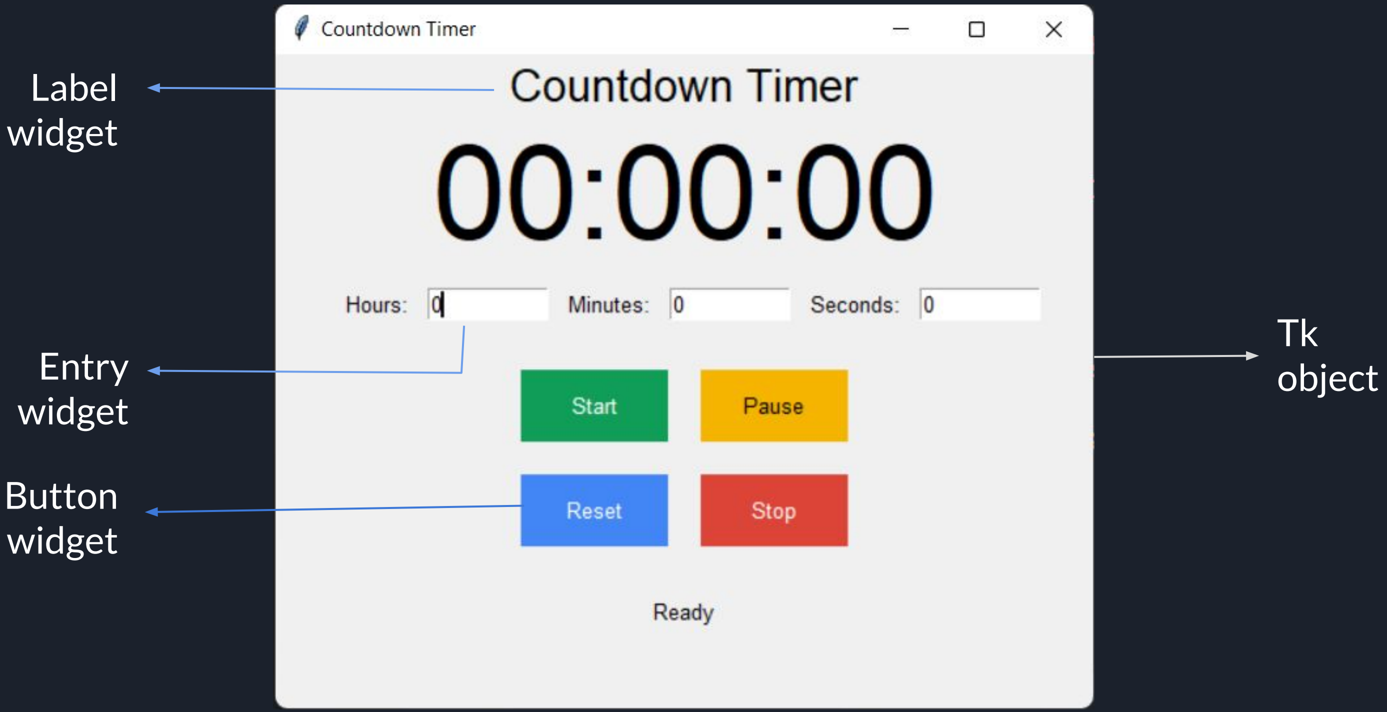
- Python 3.10.5 (64-bit)
- *tkinter* python library
- *time* module
- *threading* module

Overview



The program uses the built-in Tkinter library of python for the Graphical User Interface of the program.
It uses the *Tk* object of the tkinter library to create the main window with the size 600x400 pixels.
It also uses the different widgets provided by the *tkinter* library to render different elements of the program e.g. Label, Entry, Button widgets.

Overview



Countdown Timer

Countdown Timer

00:00:00

Hours: 0 Minutes: 0 Seconds: 0

Ready

Solution

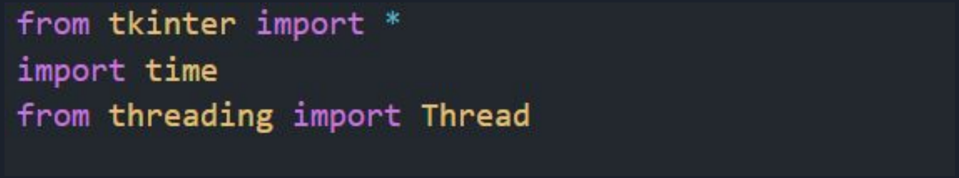
The program starts with importing the required modules/libraries i.e. *tkinter, time and threading* libraries.

A **Timer** class is created to carry out the different events of the countdown timer.

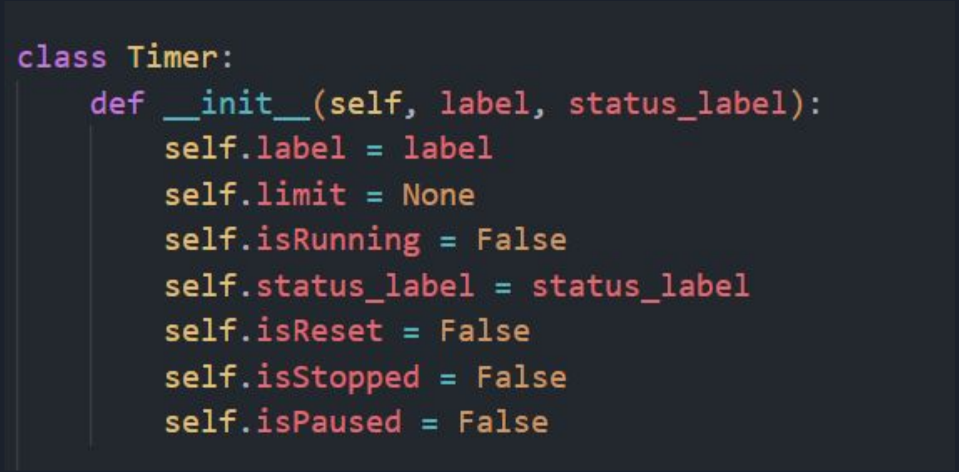
The constructor takes two required parameters i.e. *label* and *status_label*.

- *label:*
  - Type: tkinter *label* object
  - To update the countdowns.
- *Status_label:*
  - Type: tkinter *label* object
  - To update the current status of the program.

Other properties are set for the class to detect whether the program has been stopped, reset, paused or running.



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



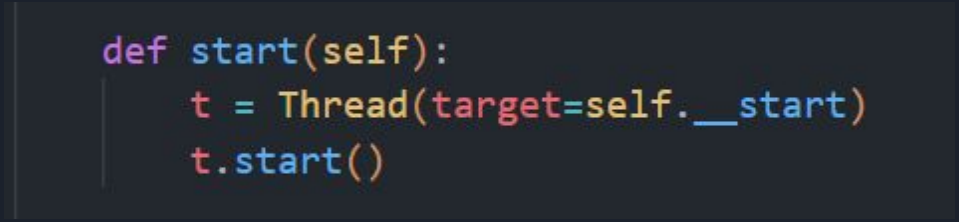
```python
class Timer:
    def __init__(self, label, status_label):
        self.label = label
        self.limit = None
        self.isRunning = False
        self.status_label = status_label
        self.isReset = False
        self.isStopped = False
        self.isPaused = False
```

Solution

Further, the Timer class also contains functions which are called by button press from Tkinter window.

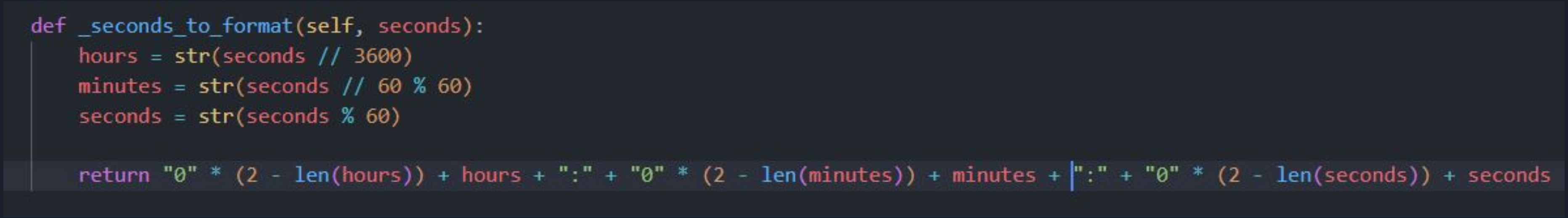
The functions include:

- *start*:
  - It takes no parameters and returns None.
  - Creates a *threading.Thread* object and calls *__start* function on different threading to avoid freezing of UI.
- *__start*:
  - It takes no parameters and returns None.
  - Starts the timer and updates the countdown label every second. *time.sleep* method is called to wait for one second. Also uses the *second_to_format* function to format the label's text in format hh:mm:ss

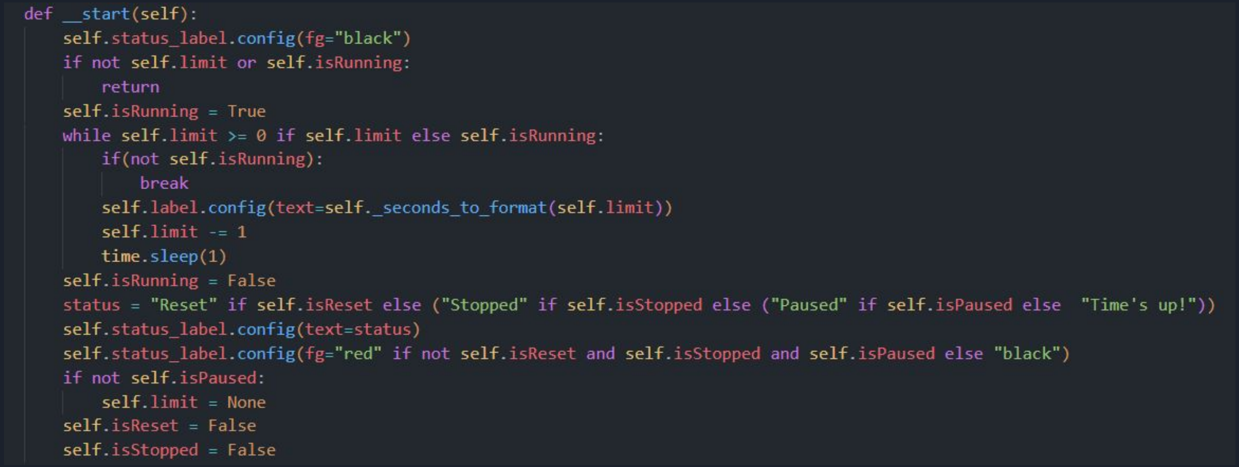


```
def start(self):
    t = Thread(target=self.__start)
    t.start()
```

Solution



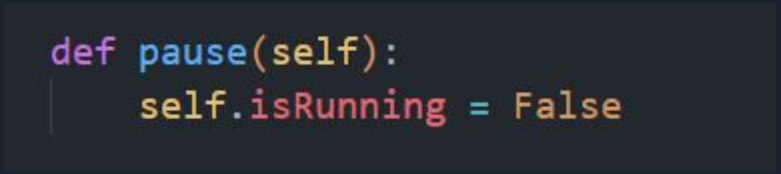
```python
def _seconds_to_format(self, seconds):
    hours = str(seconds // 3600)
    minutes = str(seconds // 60 % 60)
    seconds = str(seconds % 60)

    return "0" * (2 - len(hours)) + hours + ":" + "0" * (2 - len(minutes)) + minutes + ":" + "0" * (2 - len(seconds)) + seconds
```



```python
def __start(self):
    self.status_label.config(fg="black")
    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._seconds_to_format(self.limit))
        self.limit -= 1
        time.sleep(1)
    self.isRunning = False
    status = "Reset" if self.isReset else ("Stopped" if self.isStopped else ("Paused" if self.isPaused else  "Time's up!"))
    self.status_label.config(text=status)
    self.status_label.config(fg="red" if not self.isReset and self.isStopped and self.isPaused else "black")
    if not self.isPaused:
        self.limit = None
    self.isReset = False
    self.isStopped = False
```

Solution

- *_seconds_to_format*:
    - It takes seconds (int) as parameter and converts to hh:mm:ss format.
    - Returns the seconds value in hh:mm:ss formatted string.
- *pause:*
    - It takes no parameters and returns None.
    - Sets the isRunning property of Timer class to False so as to pause the Timer.



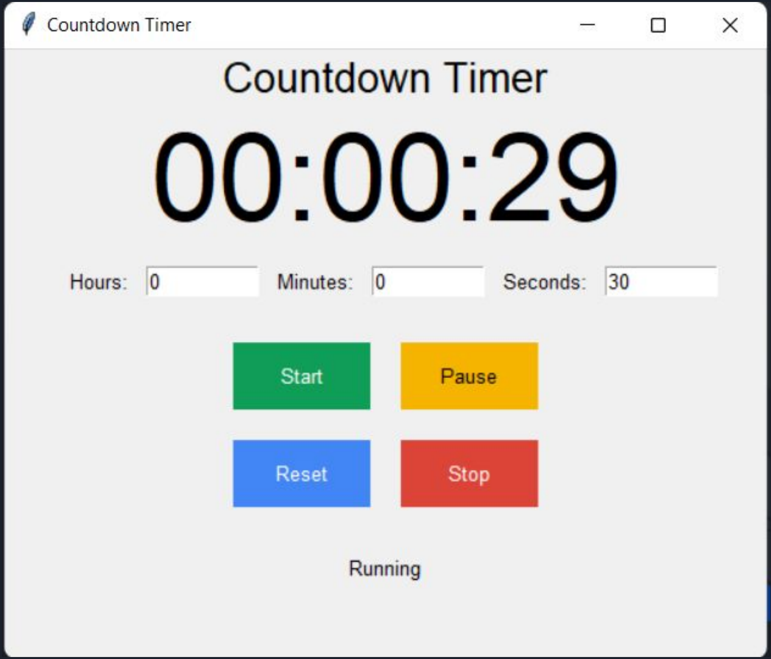
```python
def pause(self):
    self.isRunning = False
```

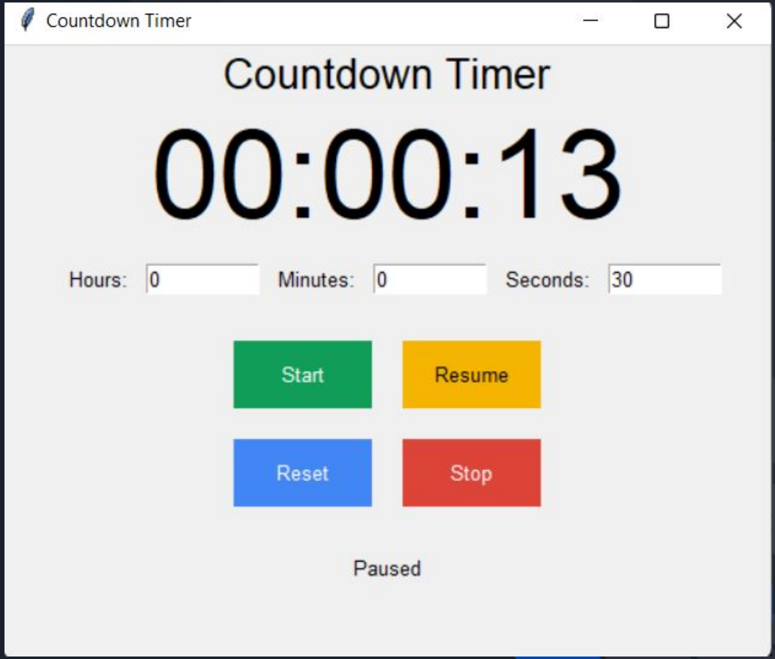
Solution

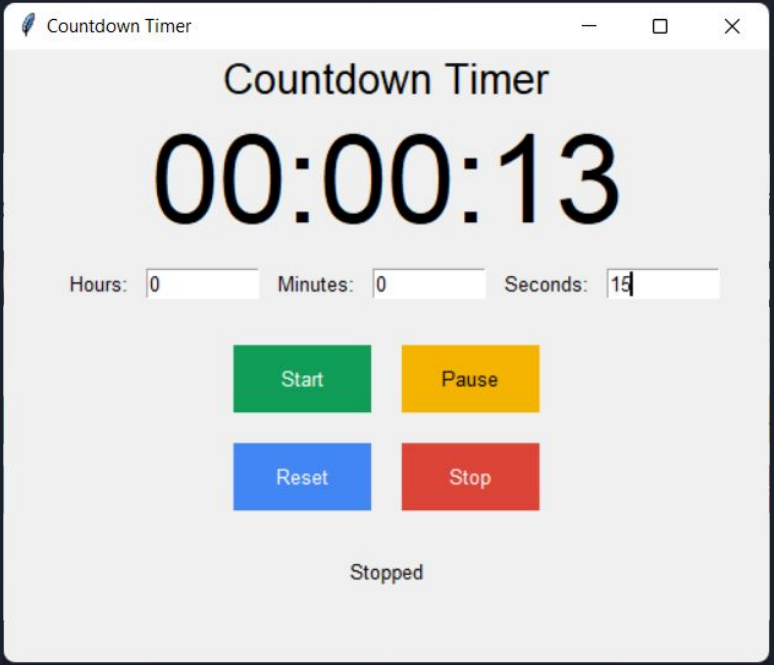
The program also uses main() function to create the window and render the different widgets over it. It also contains functions for starting, stopping, pausing, resetting and resuming the timer.

The status label in bottom is used to indicate the current state of the program.

Implementation


Countdown Timer
Countdown Timer
00:00:29
Hours: 0
Minutes: 0
Seconds: 30
Start
Pause
Reset
Stop
Running


Countdown Timer
Countdown Timer
00:00:13
Hours: 0
Minutes: 0
Seconds: 30
Start
Resume
Reset
Stop
Paused

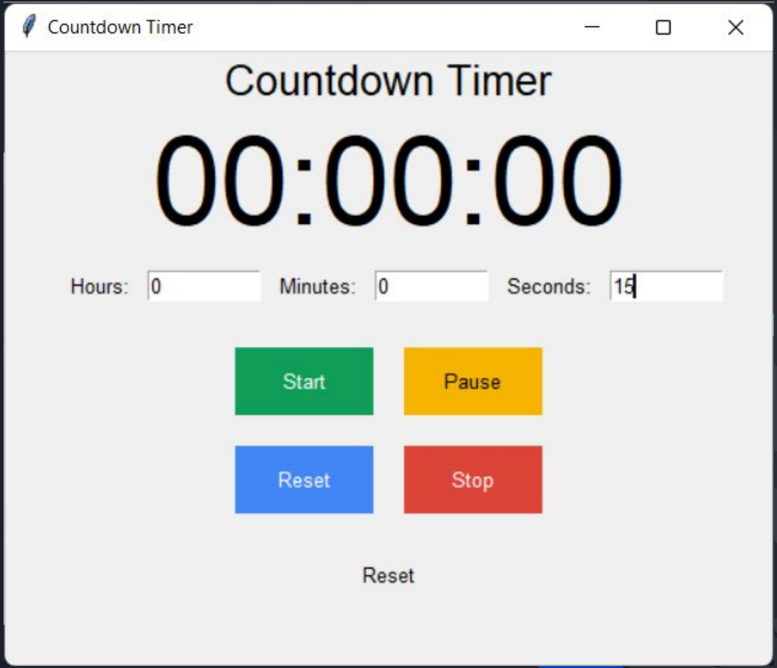

Countdown Timer

Countdown Timer

00:00:13

Hours: 0 Minutes: 0 Seconds: 15

Stopped


Countdown Timer

Countdown Timer

00:00:00

Hours: 0 Minutes: 0 Seconds: 15

Reset

References

https://www.w3schools.in/python/gui-programming

https://www.w3schools.in/python/multithreaded-programming

https://www.programiz.com/python-programming/time

Project link: https://github.com/gautamgiri-dev/countdown-timer-python