Create A Countdown Timer Using Python

Minor Project

Features To include
Reset/Stop

Pause/Resume

Sub line: AI-MINOR -AUGUST

Submitted by,
Ehill Smila J

Python Minor Project

Create A Countdown Timer Using Python

Features To Include

Reset/Stop

Pause/Resume

Introduction

Countdown Timer is used to measure whether a task is complete before the time interval or measure how much time a task has taken to complete. There are many real-life examples where it is used. For example, Online Exams, in Sports, etc.

In this tutorial, we will create a Countdown Timer in Python. We will use the Tkinter library to make it a GUI project.

You will be able to perform the following tasks:

☛Set the Time

☛Start or Resume Time

☛Pause the Countdown Time (without sleep function)

When the rest of the time is over, a music will play in the background.

The Project Details

First, you have to set the time through a Combobox, then press the 'Set' button. When the 'Set' button is pressed, two more buttons(the 'Start' button and 'Pause' button) will appear on the right side.

You can Start and Pause the timer as your need. The pause feature makes the Timer more reliable.

If you press the 'Start' button, the remaining time will show on the screen and every second will decrease by 1. When the remaining time reaches zero, a notification will be popped up with music, playing in the background.

Important note

In this project, I've used the playsound module to play an mp3 file for the notification. In the main program, you'll see a music file added like this: "Ringtones/example.mp3".

It's necessary to present this "example.mp3" file in the "Ringtones" folder in the main directory where the program file is situated. You can also use another music file; just need to mention the path properly in the program.

Requirements and Installation

- tkinter: pip install tk
- playsound: pip install playsound

Import the modules

Let's create a separate folder 'CountDown' and declare a python file there with this name: 'CountDown.py'. You need to create one more folder 'Ringtones' under the 'Countdown' directory for storing music files or ringtones there.

Declare the lists of hours and minutes

Here these two python lists contain the hours and minutes list which are gonna used in the upcoming program.

Declare the 'CountDown' class

Here all the Tkinter widgets (Labels, Buttons, Combo boxes, etc) have been declared.

The Cancel function

When the Cancel Button is pressed, this function gets a call. It stops the program and destroys the GUI window

Get the time from the user

When the 'Set' Button is pressed, this 'Get_Time()' function takes the time set by the user. If the users try to set the default value(0:0:0, see the yellow line), the program will show a warning message on the screen to set a correct time.

After pressing the 'Set' Button, the 'Start' and 'Pause' buttons appear on the right side that allows the users to Start or Resume and Pause the time as their need.

Creating an another Thread

Now create an another Thread for handling these tasks:

1. Showing the Remaining time

2. Handling Start and Pause function perfectly.

It will help to running the CountDown Timer Smoothly.

Clear Screen

Look at the 'CountDown' class (yellow lines); I've added a frame named 'button_frame' there for the Start and Pause buttons. When this function gets called, it simply clears all the widgets (only Start and Pause buttons) present there.

The Start function

See, at the very first I've set self.pause = False. In every iteration of the while loop the code checks if the status of self.pause is True or not. When it's True (when the pause button is pressed), the while loop breaks.

The Pause function

Look, at the very first the status of self.pause is set True. It pauses the time from reducing in the while loop at the Start function(remember, it happens when the while loop breaks). Then it only displays the remaining time on the screen at the pause condition.

Full Code

```
'''Create a Countdown timer in python with Start
and Pause Button'''
# Import the time module
import time
from tkinter import *
import multiprocessing
from tkinter import ttk, messagebox
from playsound import playsound
from threading import *


# Hour list
hour_list = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
15, 16, 17, 18, 19, 20, 21, 22, 23, 24]

# Minute List
min_sec_list = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29,
30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44,
45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59,
]

# Creating a CounDown Class
class CountDown:
    def __init__(self, root):
        self.window = root
        self.window.geometry("480x320+0+0")
        self.window.title('CountDown Timer')
        # Tkinter window background color
        self.window.configure(bg='gray35')
```

```
# Fixing the Window length constant
self.window.resizable(width = False, height = False)

# Declaring a variable to pause the countdown time
self.pause = False

# The Start and Pause buttons are placed
# inside this frame
self.button_frame = Frame(self.window, bg="gray35", \
width=240, height=40)
self.button_frame.place(x=230, y=150)
# This frame is used to show the countdown time label
self.time_frame = Frame(self.window, bg="gray35", \
width=480, height=120).place(x=0, y=210)

# Tkinter Labels
time_label = Label(self.window, text="Set Time",
font=("times new roman",20, "bold"), bg='gray35',fg='yellow')
time_label.place(x=180, y=30)

hour_label = Label(self.window, text="Hour",
font=("times new roman",15), bg='gray35', fg='white')
hour_label.place(x=50, y=70)

minute_label = Label(self.window, text="Minute",
font=("times new roman",15), bg='gray35', fg='white')
minute_label.place(x=200, y=70)

second_label = Label(self.window, text="Second",
font=("times new roman",15), bg='gray35', fg='white')
second_label.place(x=350, y=70)
# ==========================================

# Tkinter Comboboxes
# Combobox for hours
self.hour = IntVar()
self.hour_combobox = ttk.Combobox(self.window, width=8,
height=10, textvariable=self.hour,
font=("times new roman",15))
self.hour_combobox['values'] = hour_list
self.hour_combobox.current(0)
self.hour_combobox.place(x=50,y=110)

# Combobox for minutes
```

```
        self.minute = IntVar()
        self.minute_combobox = ttk.Combobox(self.window, width=8,
        height=10, textvariable=self.minute,
        font=("times new roman",15))
        self.minute_combobox['values'] = min_sec_list
        self.minute_combobox.current(0)
        self.minute_combobox.place(x=200,y=110)

        # Combobox for seconds
        self.second = IntVar()
        self.second_combobox = ttk.Combobox(self.window, width=8,
        height=10, textvariable=self.second,
        font=("times new roman",15))
        self.second_combobox['values'] = min_sec_list
        self.second_combobox.current(0)
        self.second_combobox.place(x=350,y=110)
        # ==========================================

        # Tkinter Buttons
        # Cancel button
        cancel_button = Button(self.window, text='Cancel',
        font=('Helvetica',12), bg="white", fg="black",
        command=self.Cancel)
        cancel_button.place(x=70, y=150)

        # Set Time Button
        # When the user will press this button
        # the 'Start' and 'Pause' button will
        # show inside the 'self.button_frame' frame
        set_button = Button(self.window, text='Set',
        font=('Helvetica',12), bg="white", fg="black",
        command=self.Get_Time)
        set_button.place(x=160, y=150)

    # It will destroy the window
    def Cancel(self):
        self.pause = True
        self.window.destroy()

    # When the set button is pressed, this
    # function gets called
    def Get_Time(self):
        self.time_display = Label(self.time_frame,
        font=('Helvetica', 20 , "bold"),
```

```
    bg = 'gray35', fg = 'yellow')
    self.time_display.place(x=130, y=210)

    try:
        # Total amount of time in seconds
        h = (int(self.hour_combobox.get())*3600)
        m = (int(self.minute_combobox.get())*60)
        s = (int(self.second_combobox.get()))
        self.time_left = h + m + s

        # If the user try to set the default time(0:0:0) then
        # a warning message will display
        if s == 0 and m == 0 and h == 0:
            messagebox.showwarning('Warning!',\
            'Please select a right time to set')
        else:
            # Start Button
            start_button = Button(self.button_frame, text='Start',
            font=('Helvetica',12), bg="green", fg="white",
            command=self.Threading)
            start_button.place(x=20, y=0)

            # Pause Button
            pause_button = Button(self.button_frame, text='Pause',
            font=('Helvetica',12), bg="red", fg="white",
            command=self.pause_time)
            pause_button.place(x=100, y=0)
    except Exception as es:
        messagebox.showerror("Error!", \
        f"Error due to {es}")

# Creating a thread to run the show_time function
def Threading(self):
    # Killing a thread through "daemon=True" isn't a good idea
    self.x = Thread(target=self.start_time, daemon=True)
    self.x.start()

# It wil clear all the widgets inside the
# 'self.button_frame' frame(Start and Pause buttons)
def Clear_Screen(self):
    for widget in self.button_frame.winfo_children():
        widget.destroy()

def pause_time(self):
```

```
    self.pause = True

    mins, secs = divmod(self.time_left, 60)
    hours = 0
    if mins > 60:
        # hour minute
        hours, mins = divmod(mins, 60)

    self.time_display.config(text=f"Time Left: {hours}: {mins}: {secs}")
    self.time_display.update()


# When the Start button will be pressed then,
# this "show_time" function will get called.
def start_time(self):
    self.pause = False
    while self.time_left > 0:
        mins, secs = divmod(self.time_left, 60)

        hours = 0
        if mins > 60:
            # hour minute
            hours, mins = divmod(mins, 60)

        self.time_display.config(text=f"Time Left: {hours}: {mins}: {secs}")
        self.time_display.update()
        # sleep function: for 1 second
        time.sleep(1)
        self.time_left = self.time_left -1
        # When the time is over, a piece of music will
        # play in the background
        if self.time_left <= 0:
            process = multiprocessing.Process(target=playsound,
            args=('Ringtones/romantic.mp3',))
            process.start()
            messagebox.showinfo('Time Over','Please ENTER to stop playing')
            process.terminate()
            # Clearing the 'self.button_frame' frame
            self.Clear_Screen()
        # if the pause button is pressed,
        # the while loop will break
        if self.pause == True:
            break
```

```
if __name__ == "__main__":
    root = Tk()
    # Creating a CountDown class object
    obj = CountDown(root)
    root.mainloop()
```