DS - MINOR – AUGUST

CREATE A COUNTDOWN TIMER USING PYTHON

'''Create a Countdown timer using python features to include

        Reset, stop, pause, resume'''

# Import the time module

import time

from tkinter import \*

import multiprocessing

from tkinter import ttk, messagebox

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("488x300+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="pink", 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='Reset',

                            font=('Helvetica', 12), bg="blue", 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='Resume',

                                      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

                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()

OUTPUT:-

