PROJECT 1 :

ALARM CLOCK AND COUNTDOWN CLOCK PROJECT

1 . INTRODUCTION

The application I've chosen is a **ALARM CLOCK AND COUNTDOWN CLOCK**. We already have a clock on our computer, of course, and probably a clock display on nearly every electronic device in our house.
CLOCK is a time management application included with Windows and Windows 10 Mobile with four key features: **alarms, world clocks, countdown clock, and a stopwatch.**
Once you've chosen a programming project, the first step is the planning stage, followed by a skeleton implementation stage, and then by a series of steps in which each part of the skeleton is fleshed out and individually tested. Of these, the planning stage is the most important; an aphorism in the programming business is "**the code that's written first is completed last".**

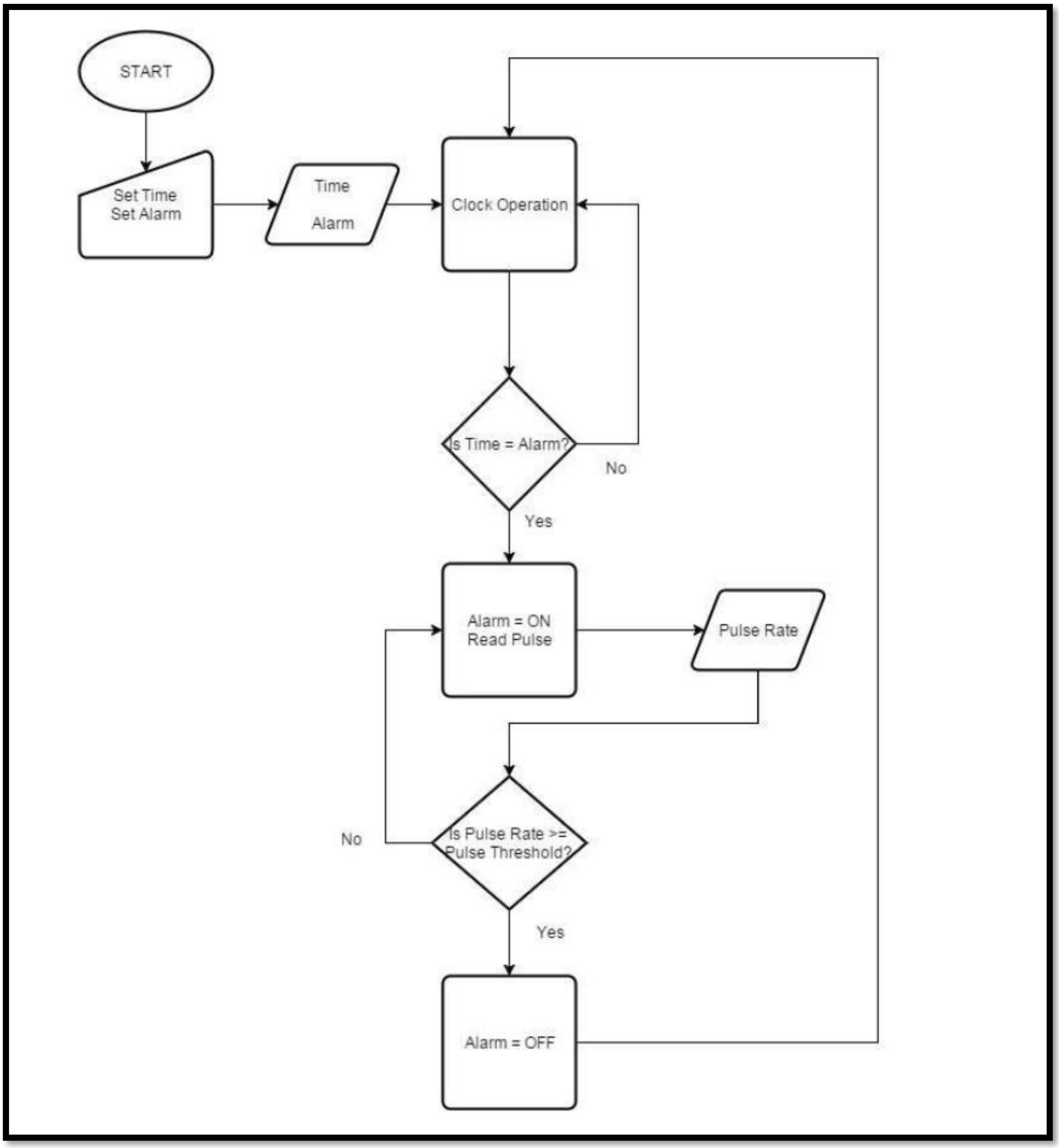
2 . ABOUT

The **alarms** are listed vertically by time of day and can be activated or deactivated with an ovular on/off switch. It is possible to delete a selection of alarms simultaneously . Alarms are triggered by a special type of toast notification for alarms. Due to hardware limitations, alarms cannot always appear on certain devices that are powered off.
A **countdown** is a sequence of backward counting to indicate the time remaining before an event is scheduled to occur.

3 . THE PLAN

For the application , we've examined all of the important system-provided tools that we'll use in future sessions: the **Tkinter** module, controls including the Canvas, Button, Label, ComboBox , Entry, Scale, the Timer, and the Random number generator. We'll work on the creation of an application using many of these facilities.
Tkinter module present in python which helps developing **GRAPHICAL USER INTERFACE**.
We used **math** module to perform mathematical function , **datetime** module for current date and **time** module for evaluating the time. The **webbrowser** module provides a high-level interface to allow displaying Web-based documents to users. Under most circumstances, simply calling the open() function from this module will open url using the default browser.

4 . FLOW CHART


START
Set Time
Set Alarm
Time
Alarm
Clock Operation
Is Time = Alarm?
No
Yes
Alarm = ON
Read Pulse
Pulse Rate
Is Pulse Rate >= Pulse Threshold?
No
Yes
Alarm = OFF

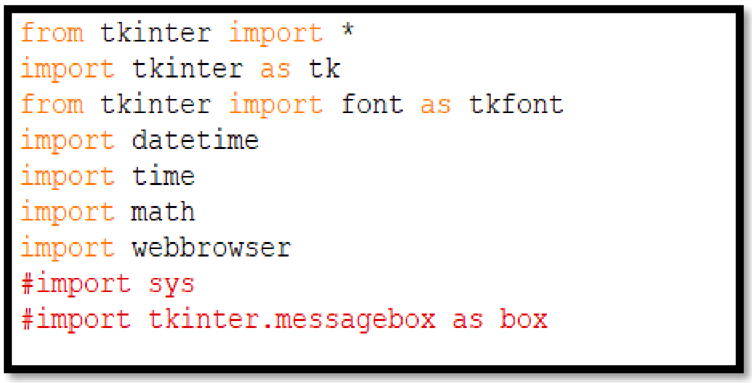
5 . CODE

The GitHub repository link of the project is:
https://github.com/Disha6/Alarm-Clock-and-Countdown-Clock

6 . EXPLANATION

Here, we'll see about the code block by block and see the detailed information about each line to make the project .

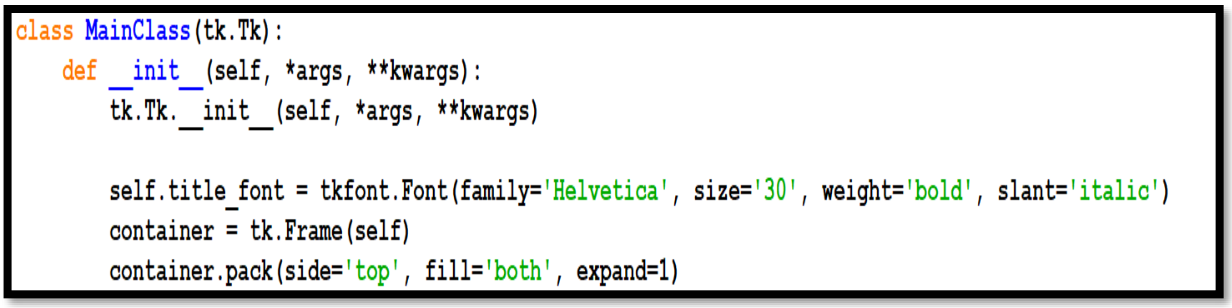
a . Importing Modules – The very first step in order to create our project is to import all the modules required .



```
from tkinter import *
import tkinter as tk
from tkinter import font as tkfont
import datetime
import time
import math
import webbrowser
#import sys
#import tkinter.messagebox as box
```

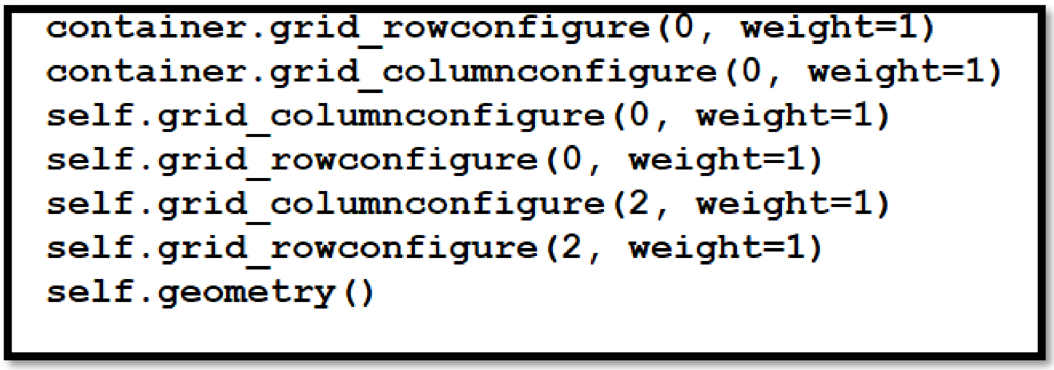
b . Layout Designing – Here , this section contains what user see on the window of the application.

The layout design initiates with basic window design which includes the title , font (size ,family , weight ,frame , slant etc.).



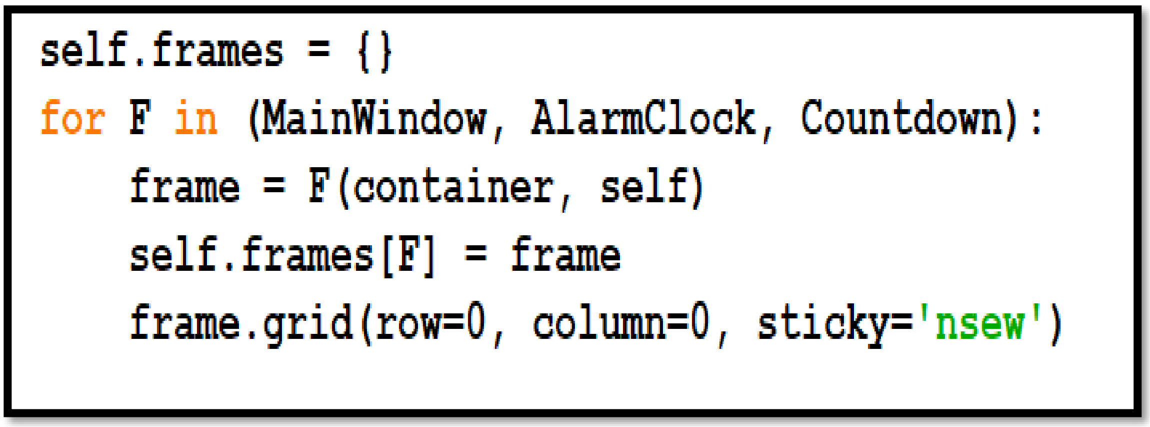
```
class MainClass(tk.Tk):
    def __init__(self, *args, **kwargs):
        tk.Tk.__init__(self, *args, **kwargs)

        self.title_font = tkfont.Font(family='Helvetica', size='30', weight='bold', slant='italic')
        container = tk.Frame(self)
        container.pack(side='top', fill='both', expand=1)
```

The below line then tells about the row configuration , column configuration and the whole geometry of the front end which appears on the screen of the user.



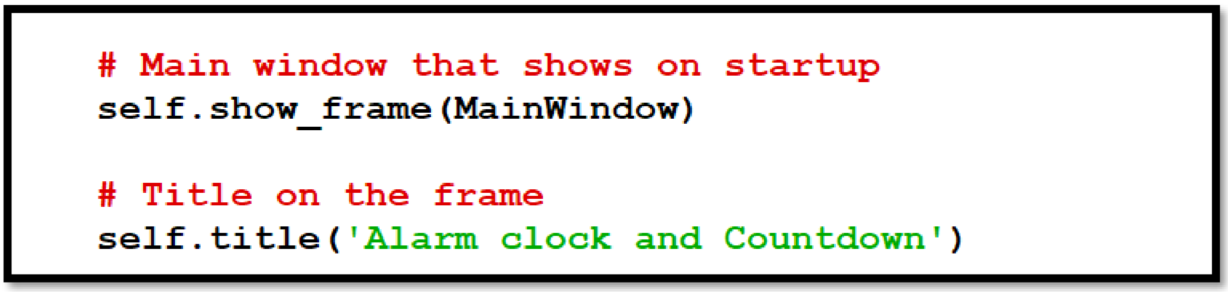
```
container.grid_rowconfigure(0, weight=1)
container.grid_columnconfigure(0, weight=1)
self.grid_columnconfigure(0, weight=1)
self.grid_rowconfigure(0, weight=1)
self.grid_columnconfigure(2, weight=1)
self.grid_rowconfigure(2, weight=1)
self.geometry()
```

This is the code that makes it possible to switch between windows in the same frame .



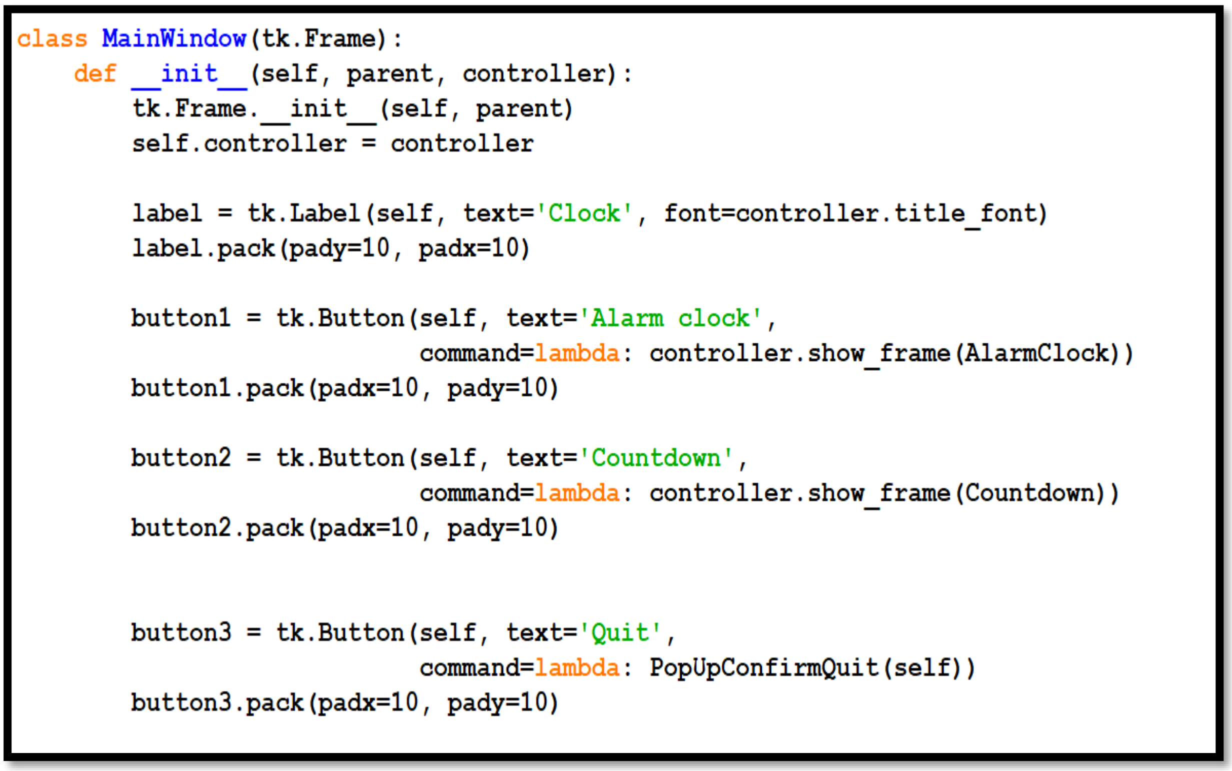
```
self.frames = {}
for F in (MainWindow, AlarmClock, Countdown):
    frame = F(container, self)
    self.frames[F] = frame
    frame.grid(row=0, column=0, sticky='nsew')
```

From the code below , it shows the min window and title which appears on the window.



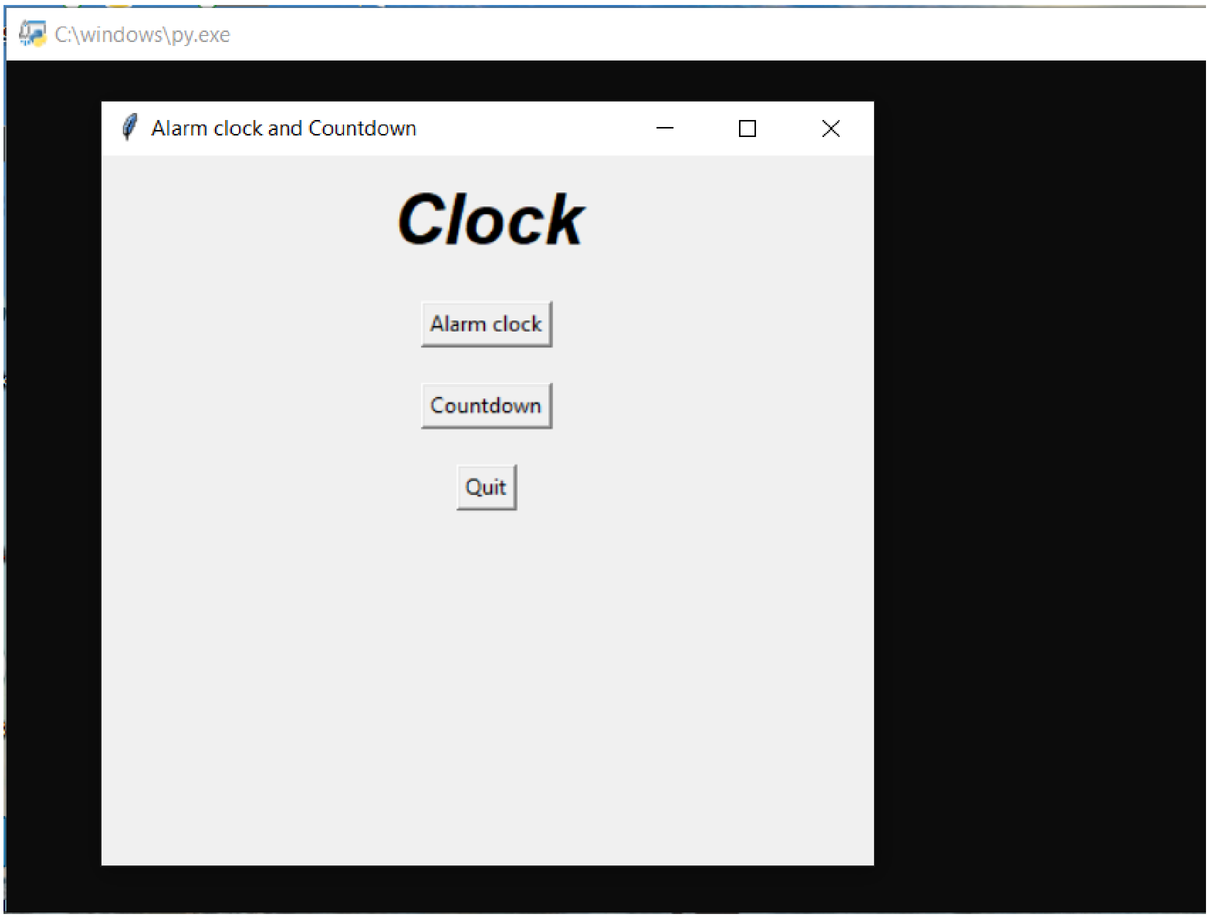
```
# Main window that shows on startup
self.show_frame(MainWindow)

# Title on the frame
self.title('Alarm clock and Countdown')
```

Now , with the help of controller and labels we will make the button so that user can choose **ALARM CLOCK OR COUNTDOWN** as per use and if user wants to **QUIT** the application that option is also available on the window.



```
class MainWindow(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller

        label = tk.Label(self, text='Clock', font=controller.title_font)
        label.pack(pady=10, padx=10)

        button1 = tk.Button(self, text='Alarm clock',
                            command=lambda: controller.show_frame(AlarmClock))
        button1.pack(padx=10, pady=10)

        button2 = tk.Button(self, text='Countdown',
                            command=lambda: controller.show_frame(Countdown))
        button2.pack(padx=10, pady=10)


        button3 = tk.Button(self, text='Quit',
                            command=lambda: PopUpConfirmQuit(self))
        button3.pack(padx=10, pady=10)
```

This is how the main window will looks to the user who we work on it .



From here the coding of the alarm clock starts . The current time will be displayed on window . User have to give input of the time(in hours and minutes) when they need alarm to rings. When the current time is equal to the time input by the user then the clock works and rings the bell .

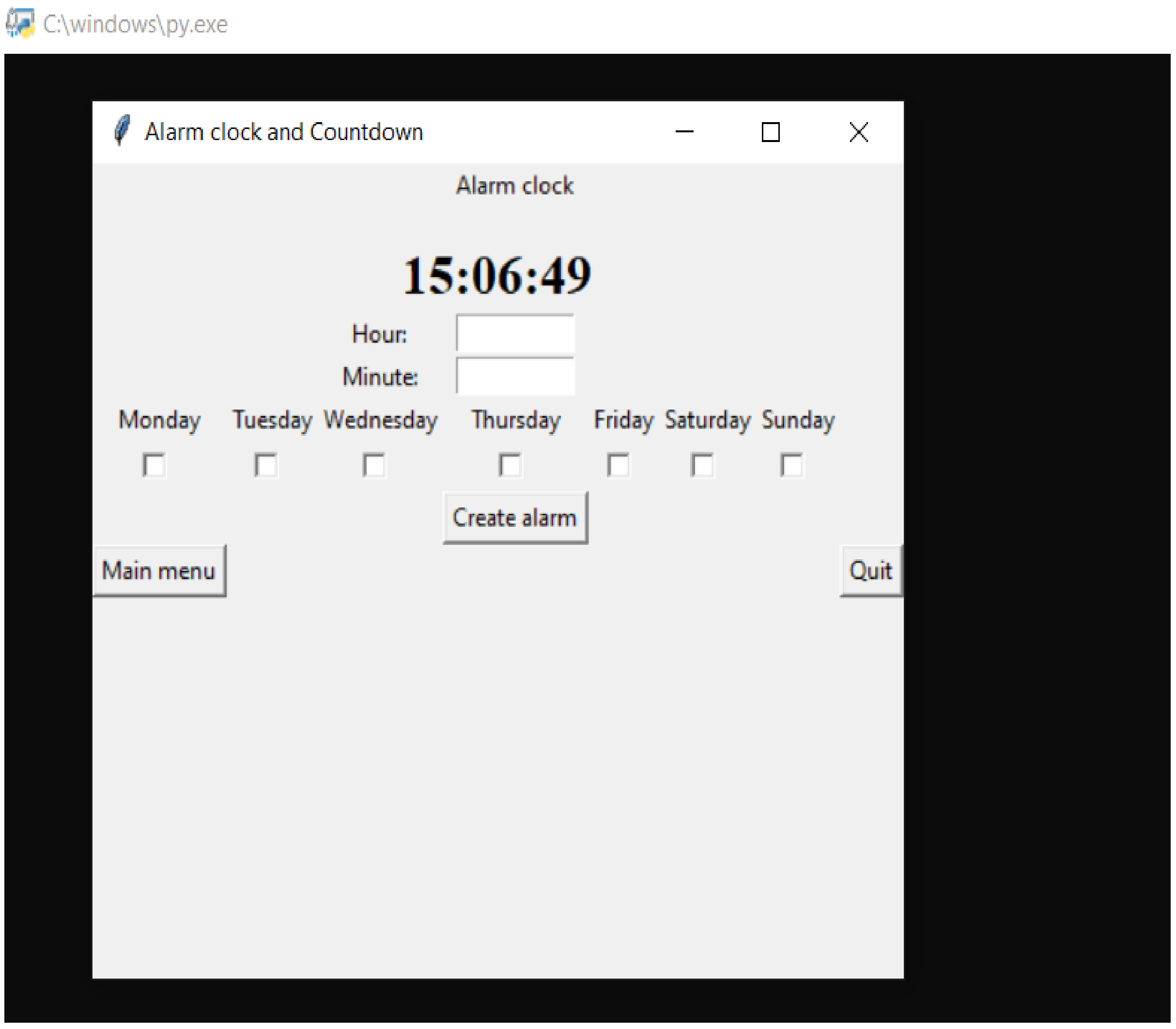
The box is made so that user can give input. All the commands are wriiten before the input box so that user need not to think much about it and waste time.

```python
class AlarmClock(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller

        # self.hour = hour

        tk.Label(self, text='Alarm clock\n').grid(row=1, column=4)

        current_time = tk.Label(self, font=('times', 20, 'bold'))
        current_time.grid(row=2, columns=12)

        tk.Label(self, text='Hour:').grid(row=3, column=3)

        alarm_hour_entry = Entry(self, width=10)
        alarm_hour_entry.grid(row=3, column=4)

        tk.Label(self, text='Minute:').grid(row=4, column=3)

        alarm_minute_entry = Entry(self, width=10)
        alarm_minute_entry.grid(row=4, column=4)
```

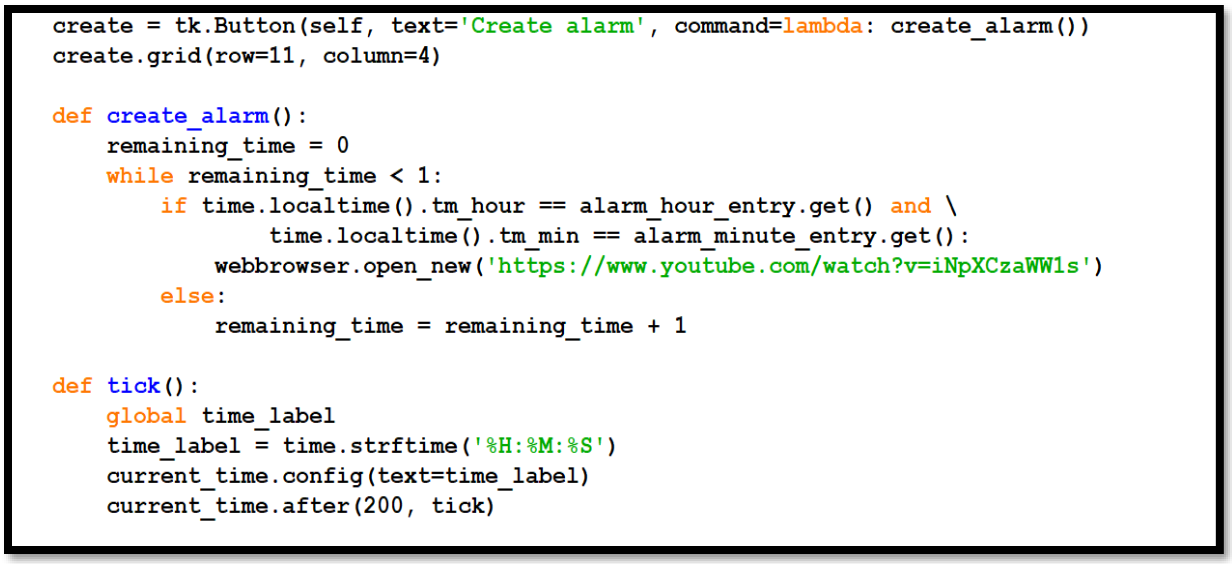
We can also choose the day.

```python
tk.Label(self, text='Monday').grid(row=7, column=1)
tk.Label(self, text='Tuesday').grid(row=7, column=2)
tk.Label(self, text='Wednesday').grid(row=7, column=3)
tk.Label(self, text='Thursday').grid(row=7, column=4)
tk.Label(self, text='Friday').grid(row=7, column=5)
tk.Label(self, text='Saturday').grid(row=7, column=6)
tk.Label(self, text='Sunday').grid(row=7, column=7)

monday_box = tk.Checkbutton(self)
monday_box.grid(row=10, column=1)

tuesday_box = tk.Checkbutton(self)
tuesday_box.grid(row=10, column=2)

wednesday_box = tk.Checkbutton(self)
wednesday_box.grid(row=10, column=3)

thursday_box = tk.Checkbutton(self)
thursday_box.grid(row=10, column=4)

friday_box = tk.Checkbutton(self)
friday_box.grid(row=10, column=5)

saturday_box = tk.Checkbutton(self)
saturday_box.grid(row=10, column=6)

sunday_box = tk.Checkbutton(self)
sunday_box.grid(row=10, column=7)
```

View of the main window.

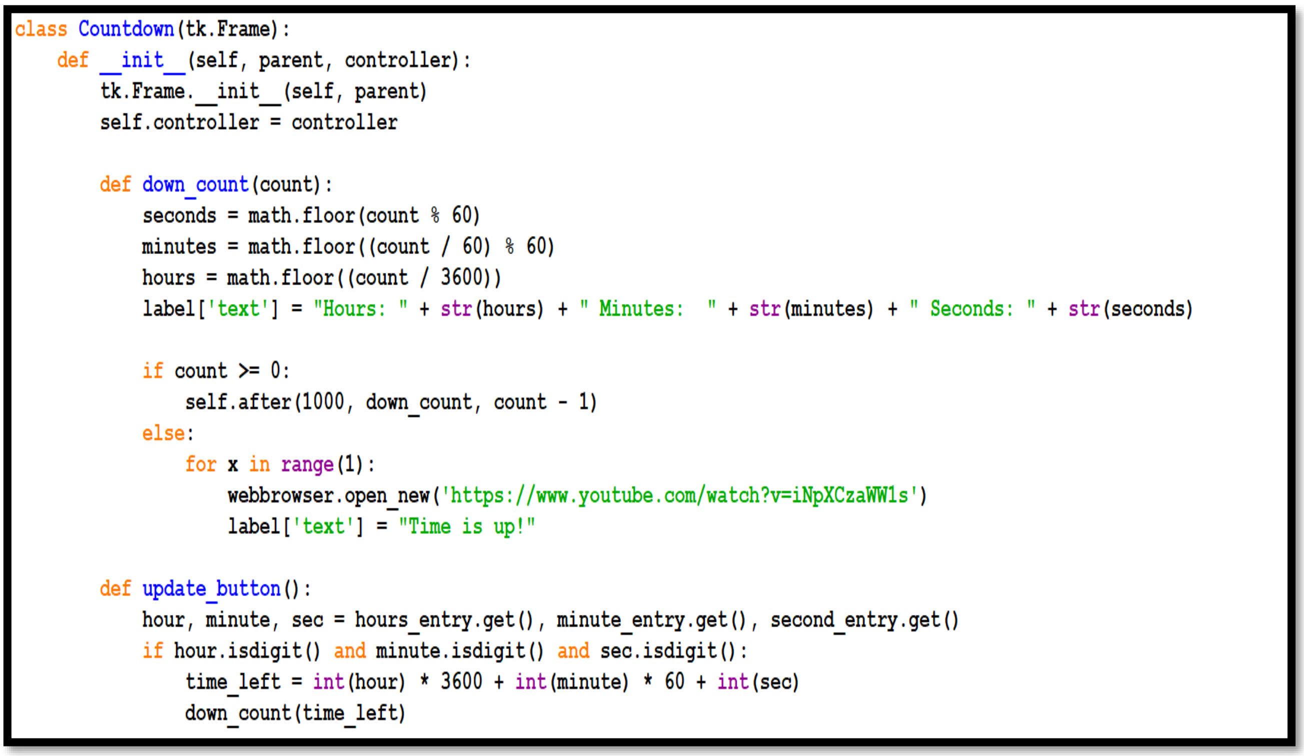


This is the working of alarm clock that how when the time is reached the alarm rings . Here we see the use of web browser so to link any youtube song , voice etc which rings at the moment.



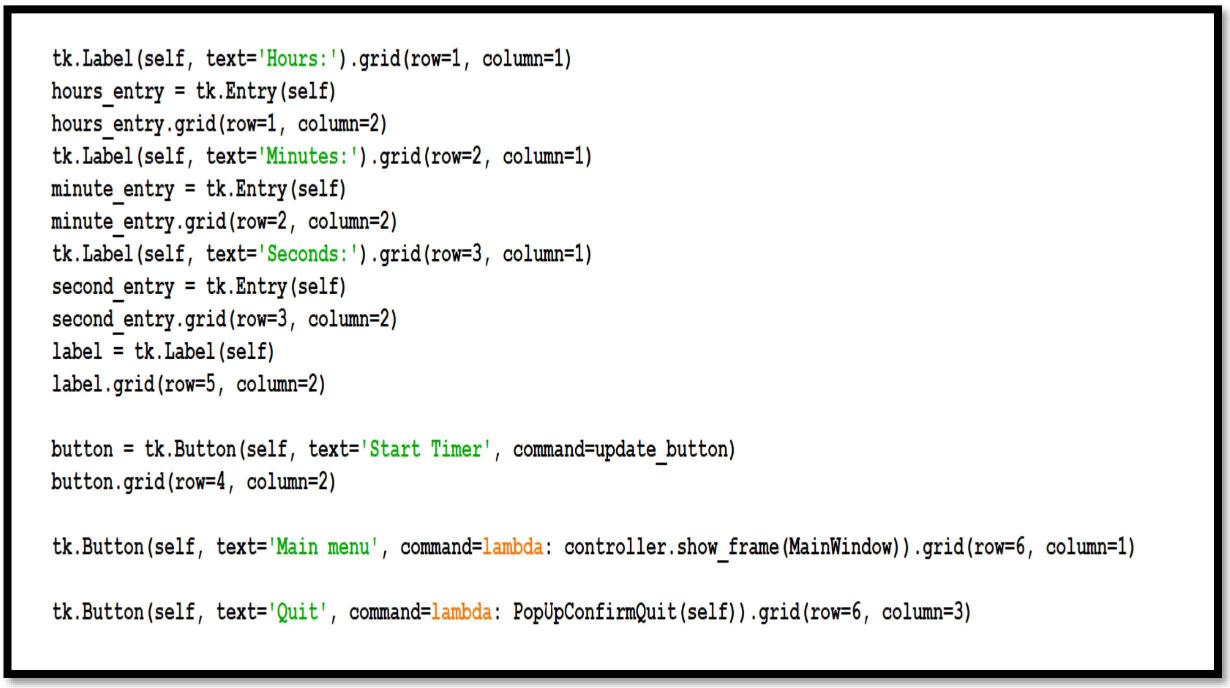
```python
create = tk.Button(self, text='Create alarm', command=lambda: create_alarm())
create.grid(row=11, column=4)

def create_alarm():
    remaining_time = 0
    while remaining_time < 1:
        if time.localtime().tm_hour == alarm_hour_entry.get() and \
                time.localtime().tm_min == alarm_minute_entry.get():
            webbrowser.open_new('https://www.youtube.com/watch?v=iNpXCzaWW1s')
        else:
            remaining_time = remaining_time + 1

def tick():
    global time_label
    time_label = time.strftime('%H:%M:%S')
    current_time.config(text=time_label)
    current_time.after(200, tick)
```

From here starts the working of countdown clock .



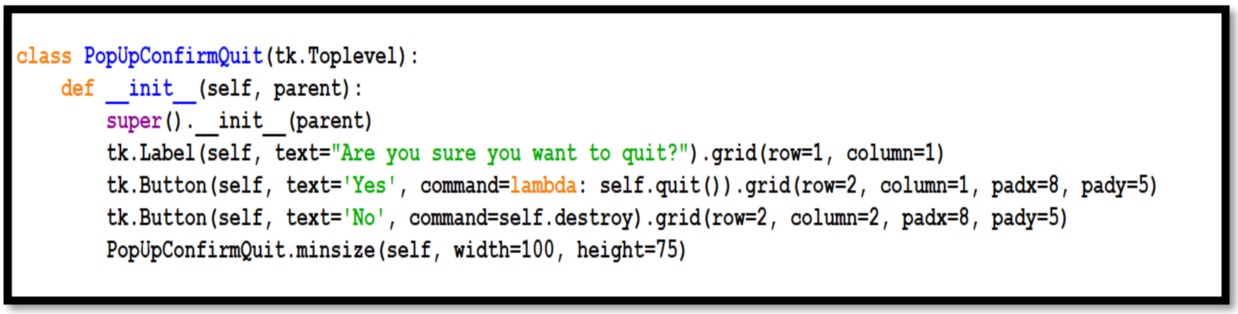
```python
class Countdown(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller

        def down_count(count):
            seconds = math.floor(count % 60)
            minutes = math.floor((count / 60) % 60)
            hours = math.floor((count / 3600))
            label['text'] = "Hours:  " + str(hours) + " Minutes:  " + str(minutes) + " Seconds:  " + str(seconds)

            if count >= 0:
                self.after(1000, down_count, count - 1)
            else:
                for x in range(1):
                    webbrowser.open_new('https://www.youtube.com/watch?v=iNpXCzaWW1s')
                    label['text'] = "Time is up!"

        def update_button():
            hour, minute, sec = hours_entry.get(), minute_entry.get(), second_entry.get()
            if hour.isdigit() and minute.isdigit() and sec.isdigit():
                time_left = int(hour) * 3600 + int(minute) * 60 + int(sec)
                down_count(time_left)
```

We can set the hours , minutes and seconds left and a alarm rings and a message is displayed on a screen that **“Time is up!” .** We can choose the **MAIN MENU** option to go to starting window or can even **QUIT**.



```
tk.Label(self, text='Hours:').grid(row=1, column=1)
hours_entry = tk.Entry(self)
hours_entry.grid(row=1, column=2)
tk.Label(self, text='Minutes:').grid(row=2, column=1)
minute_entry = tk.Entry(self)
minute_entry.grid(row=2, column=2)
tk.Label(self, text='Seconds:').grid(row=3, column=1)
second_entry = tk.Entry(self)
second_entry.grid(row=3, column=2)
label = tk.Label(self)
label.grid(row=5, column=2)

button = tk.Button(self, text='Start Timer', command=update_button)
button.grid(row=4, column=2)

tk.Button(self, text='Main menu', command=lambda: controller.show_frame(MainWindow)).grid(row=6, column=1)

tk.Button(self, text='Quit', command=lambda: PopUpConfirmQuit(self)).grid(row=6, column=3)
```

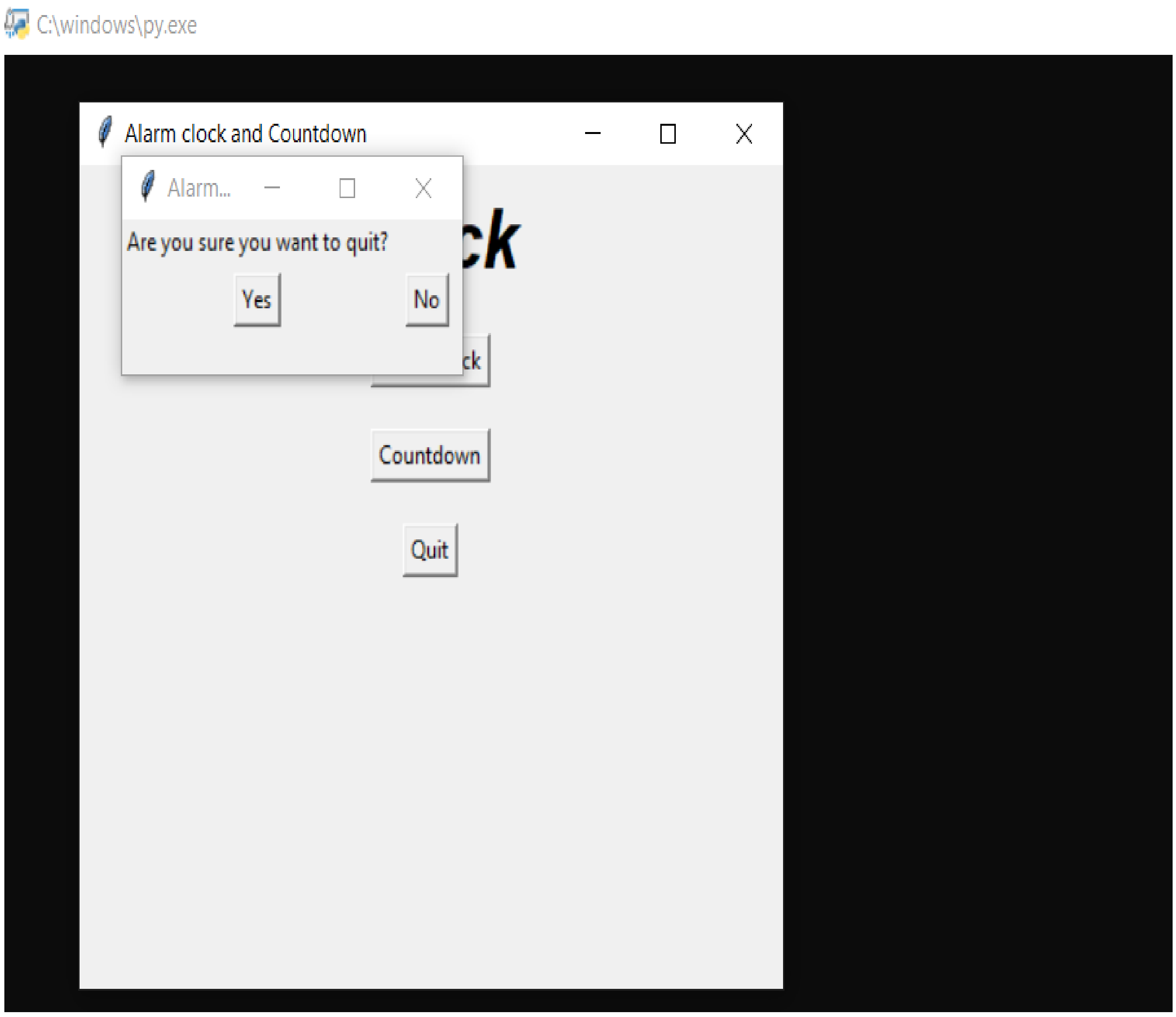
Now we come to the last part of the coding i.e, the code for quiting . Select the option **YES** if want to QUIT otherwise **NO.**



```
class PopUpConfirmQuit(tk.Toplevel):
    def __init__(self, parent):
        super().__init__(parent)
        tk.Label(self, text="Are you sure you want to quit?").grid(row=1, column=1)
        tk.Button(self, text='Yes', command=lambda: self.quit()).grid(row=2, column=1, padx=8, pady=5)
        tk.Button(self, text='No', command=self.destroy).grid(row=2, column=2, padx=8, pady=5)
        PopUpConfirmQuit.minsize(self, width=100, height=75)
```

How the final window look likes**.**

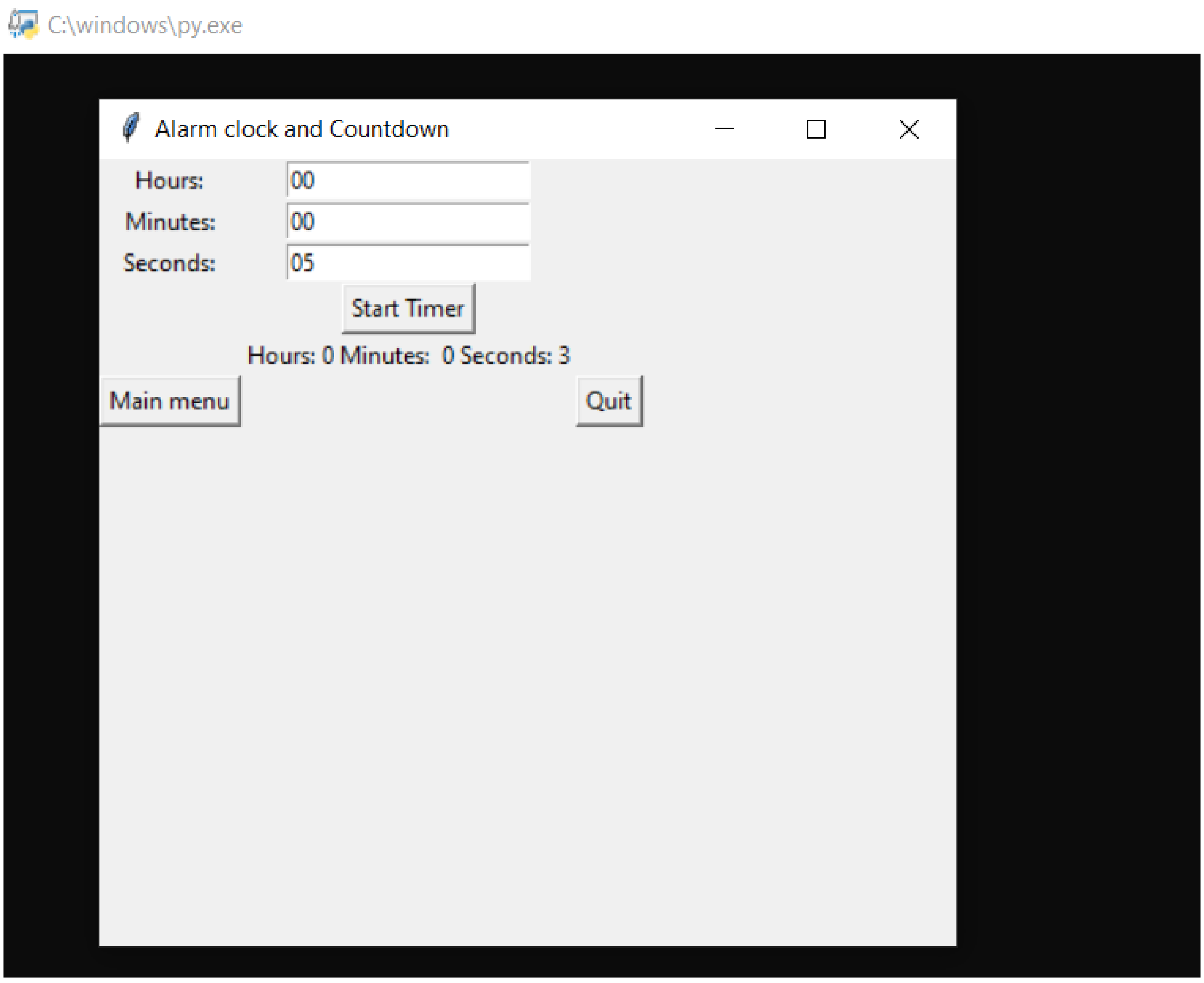
If the user want to continue they will click on **NO .** By this they will remain on the same window , and if they are willing to close the program then they click on **YES** .



Are you sure you want to quit?

Yes No

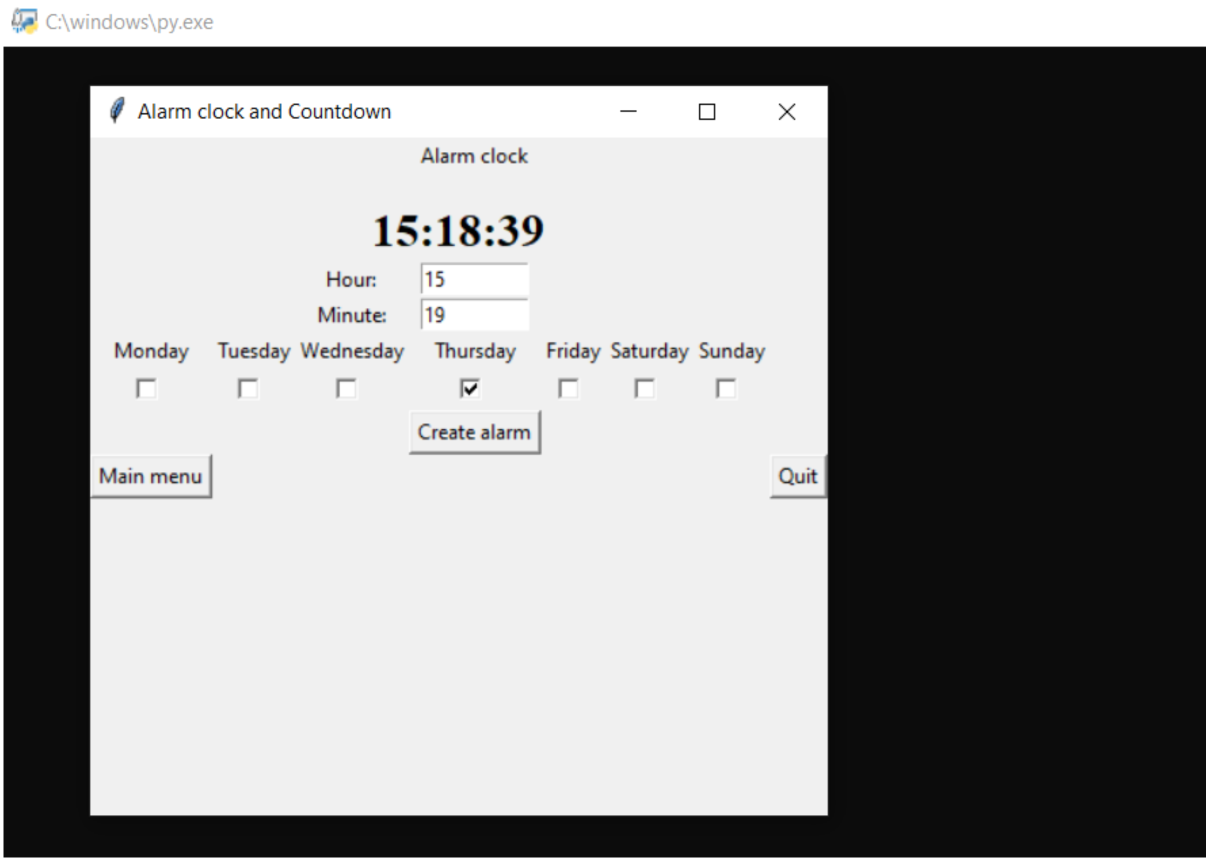
FEW OUTPUTS:


C:\windows\py.exe

Alarm clock and Countdown

```
Hours:     00
Minutes:   00
Seconds:   05
```

```
Hours: 0 Minutes:  0 Seconds: 3
```


C:\windows\py.exe

Alarm clock and Countdown

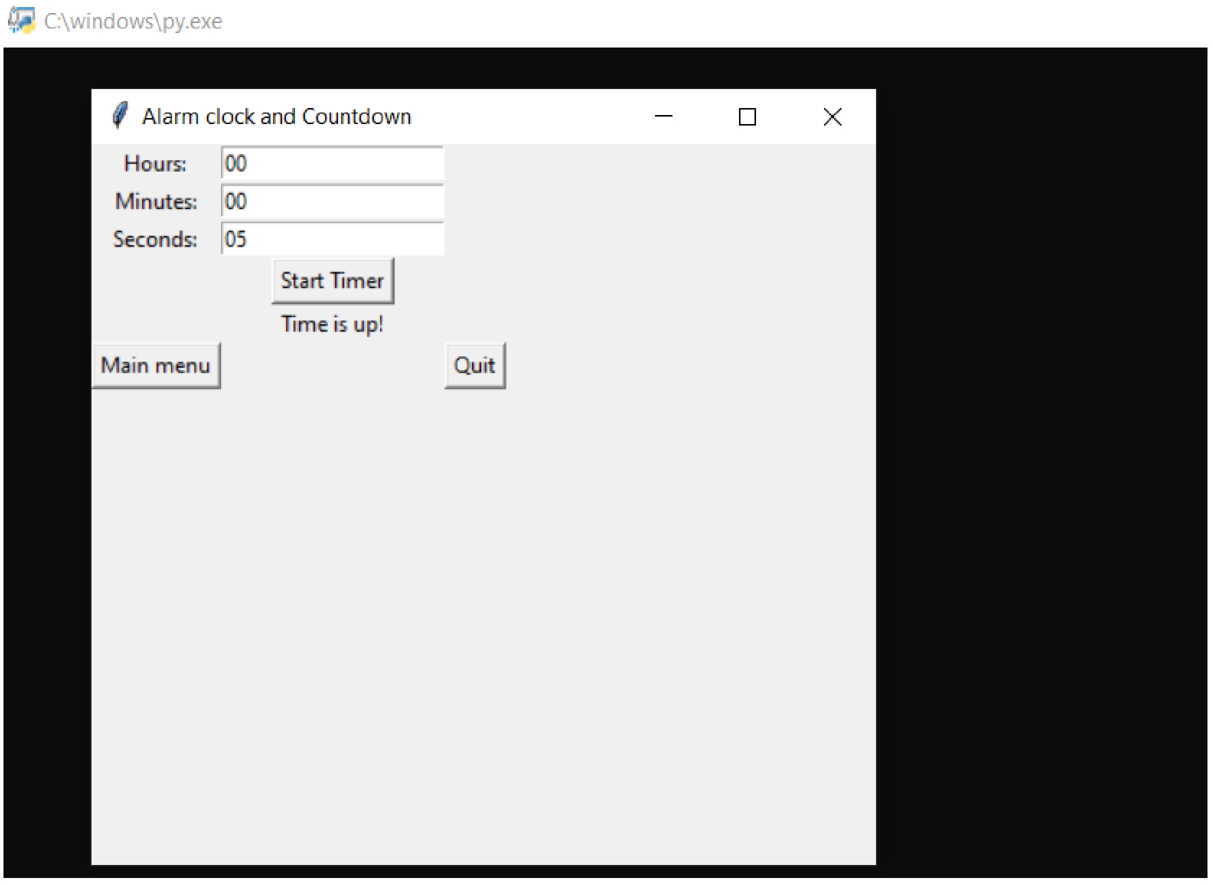
Alarm clock

15:18:39

Hour: 15

Minute: 19

Monday [ ] Tuesday [ ] Wednesday [ ] Thursday [✓] Friday [ ] Saturday [ ] Sunday [ ]


C:\windows\py.exe

Alarm clock and Countdown

Hours: 00

Minutes: 00

Seconds: 05

Time is up!