# Countdown Clock and Timer - DataFlair

#### **About Countdown Clock and Timer:**

A countdown timer is an ideal way to set targets to complete tasks, keep a reminder for the oven etc and for large scale applications such as industries, complex timers are used which are custom designed for various purposes such as triggering or flipping of a switch after a delay of set time. We might have noticed a screensaver appearing on the desktop when left idle for sometime. This happens because of the timer which calculates our idle time. Similarly automatic logging out of websites, OTP expiration, captcha expiration are all based on a timer which nullifies these password after the time period expires

#### **Python Countdown Clock and Timer Project:**

We will create a simple countdown timer using python and display 2 notifications, one with the app created and another on the desktop to remind the user of the elapse. Good understanding of functions and tkinter widgets to understand the code flow is ideal.

#### **Project Prerequisites:**

The project makes use of tkinter for GUI, time module for creating a delay and plyer to create desktop notifications. We make use of Tkinter, a built-in GUI library in python. To check its availability, import it

python

>>> import tkinter

If the library is installed, it will not show an error. In case an error appears, use the following command to install Tkinter on a linux system: sudo apt-get install python3-tkinter. On Windows systems, reinstalling python will fix the issue. Time module is a built-in library and hence does not require any installation. Install plyer using pip: pip install plyer

#### **Download Countdown Clock and Timer code:**

You can download the source code for the Countdown Clock and Timer in the given link: [Countdown clock and timer](https://drive.google.com/file/d/1DeVDmBTsIMylnoRyZr3i8cJnYFmjBjl5/view?usp=sharing)

#### **Project File Structure:**

There are many GUI libraries supported by python such as PyQT5, Kivy, Pyside2 etc. Tkinter is widely used by many developers and is easy for beginners to practise.

Let’s have a look at the steps to build the project:

1. Importing modules: time, tkinter and plyer
2. Initialising the window and declaring the dimensions
3. Defining functions for timer and placeholders
4. Creating the user input interface
5. Addition of a button to activate the timer

Now for the implementation in detail.

Feel free to play with the values and change your input methods.

##### **Importing necessary modules:**

#DataFlair Guide for Countdown Clock and Timer

#Import necessary modules

from plyer import notification

from tkinter import messagebox

from tkinter import \*

import time

**Code Explanation:**

* **from plyer import notification:**  Plyer is a library that provides us with features to access bluetooth, wifi, battery details, send emails, gps and so on. Here we make use of notification to provide desktop notifications regardless of the app in use
* **from tkinter import messagebox:** To display prompts, we use messagebox. It can create messageboxes for errors, to display information, ask questions etc.
* **From tkinter import \*:**  We make use of Tkinter to create the user interface for the application. Tkinter is suitable for beginners and contains many widgets.
* **Import time:** To create a delay between every second, we use time module

##### **Initialising the window and declaring the dimensions:**

#Assign class and set dimensions of the interface

window = Tk()

window.geometry("300x200")

window.title("DataFlair-Countdown timer and notification")

**Code explanation:**

* **Window = Tk():** Tkinter class, Tk() is assigned to an object, window. We use the object to set components on the object such as text boxes, buttons etc
* **window.geometry():**  Specify the length and the breadth of the application window of the application using geometry.
* **window.title():**  Specify the title of the application. It is optional

##### **Defining functions for timer and placeholders:**

#Remove the placeholders for every entry field based on click

def h\_click(event):

hour\_entry.delete(0, 'end')

def m\_click(event):

min\_entry.delete(0, 'end')

def s\_click(event):

sec\_entry.delete(0, 'end')

**Code explanation:**

The placeholder functions are optional.

* **def h\_click(event):** We add a placeholder in the hours, minutes and seconds entry widgets. Hence when the user selects an entry widget, the placeholder is removed by deleting using entry\_widget.entry(0,’end’), where 0 is the start of the placeholder character and ‘end’ denotes the end of the placeholder string. Similarly we extend this for minutes and seconds entry widget mouse clicks

#Function to activate timer and show notifications once timer is up

def timer():

#Since we use placeholders, we check if the user entered an integer

try:

timer\_time = int(hour\_entry.get())\*3600 + int(min\_entry.get())\*60 + int(sec\_entry.get())

except:

messagebox.showerror(message="Enter Valid Time")

#The user cannot activate a timer with no time set

#To update the timer with every decreasing second and display a notification

if timer\_time >0:

hour = 0

min = 0

sec = 0

#If minutes is more than 60, it has to be set to the next hour

while timer\_time >= 0:

min, sec = divmod(timer\_time,60)

if min > 60:

hour, min = divmod(min,60)

#Set the declared variables with the new values to display

hours.set(hour)

mins.set(min)

secs.set(sec)

#Sleep for 1 creates a delay of 1 second

time.sleep(1)

#Update the changes on the window for every second

window.update()

#Decrement the timer value by 1

timer\_time -= 1

#Create a desktop notification

notification.notify(

#Title of the notification,

title = "TIMER ALERT",

#Body of the notification

message = "Hey amigo!\nDid you do what you wanted to achieve? \nIf not, try again with a new timer",

app\_icon="/home/deepika/Downloads/internship/countdown\_timer/pictures/bell.ico",

#Notification stays for 30 seconds

timeout = 30,

)

#This notification is provided by tkinter with the created app

messagebox.showinfo(message="Timer Complete!")

**Code explanation:**

* **def show\_entry\_fields():** Declaration of function to invoke the timer.
* **timer\_time:** Set timer variable to contain the duration of the user timer in seconds. An hour contains 3600 seconds and a minute contains 60 seconds respectively. Thus we multiply hours with 3600 and minutes with 60, sum them up with the seconds
* **try...except :** If a user enters an integer, the try block will not pass into the except block. This is because we set the hour, minute and second variables to integer data type. If the user gives a float number or it is left empty, a warning will pop up
* **If timer\_time > 0:** A timer can start only when the user enters a minimum of 1 second
* **hour =0, min =0, sec =0:** Declaration and initialisation of variables
* **while timer\_time >= 0:** The loop decrements timer\_time until it reaches 0 which marks the end of the time delay.
* **min, sec:** Converting timer\_time to minutes and seconds to display on the app. divmod(value, divisor) is a function that returns 2 values. The first value is the quotient of the division of timer\_time by 60 and the second value is the remainder obtained during the division.
* **if min > 60:** If minutes is greater than 60, it is converted to hours and minutes using **hour, min = divmod(min,60)**
* **hours.set(hour), mins.set(min), secs.set(sec):** Set the obtained hours, minutes and seconds to the integer variables declared outside the function.
* **time.sleep(1):** Creates a delay of 1 second
* **timer\_time -= 1:** Decrement the timer\_time by 1 to simulate the countdown timer
* **notification.notify():** Creates a desktop notification. It takes the parameters: title - title of the alert, message - message of the notification, app\_icon - (optional), a picture or icon, timeout - the duration after which the notification elapses. Replace the path for app\_icon with your icon’s absolute path.
* **messagebox.showinfo():**  A prompt box which serves as a notification in the timer app.

##### **Creating the user input interface:**

#Label for displaying the title of the app

#position of the label or widget is set using pack().

#pack defaults to centered alignment on a x row and y column coordinate

title\_label\_1=Label(window,text="DataFlair-Countdown timer with notification", font=("Gayathri", 12)).pack()

title\_label\_2 = Label(window, text="Put 0 in fields not of use", font=("Gayathri", 10)).pack()

#Variables using which the timer is updated in the function

hours = IntVar()

mins = IntVar()

secs = IntVar()

#To read user input for hours, minutes and seconds

hour\_entry=Entry(window,width=3,textvariable=hours,font=("Ubuntu Mono",18))

min\_entry=Entry(window,width=3,textvariable=mins,font=("Ubuntu Mono",18))

sec\_entry=Entry(window,width=3,textvariable=secs,font=("Ubuntu Mono",18))

#Placeholder for the entry widgets

hour\_entry.insert(0,00)

min\_entry.insert(0,00)

sec\_entry.insert(0,00)

#Positioning the entry widgets.

#place() takes an x(from the left) and y(from the top) coordinate

hour\_entry.place(x=80,y=40)

min\_entry.place(x=130,y=40)

sec\_entry.place(x=180,y=40)

#To link the defined placeholder removal functions on mouse click

hour\_entry.bind("<1>", h\_click)

min\_entry.bind("<1>", m\_click)

sec\_entry.bind("<1>", s\_click)

**Code explanation:**

* **title\_label:** To display non editable text, we use labels. Label() has 2 mandatory parameters: window and text to display. Position the label or any widget using pack().
* **IntVar():** Declaring integer variables for hours, minutes and seconds. A prompt, in the try..except of the timer function, is raised if the user gives decimals or other characters.
* **Entry():** To take user input, we use Entry widget. The parameters are window, width of the entry field, textvariable which denotes the integer variable declared previously. Font styling is optional.
* **hour\_entry.insert():** To insert a placeholder of 00 starting with the index 0. We extend the same for minutes and seconds entry widgets.
* **hour\_entry.place():** Similar to pack(), place() is another positioning function. This function takes 2 parameters which is the distances from the left margin and the top margin as coordinates (x,y)
* **hour\_entry.bind():** Removal of the placeholder when the mouse clicks on the entry fields. <1> indicates mouse button 1 which is the left click.

##### **Addition of a button to activate the timer:**

#button to activate the timer function

button = Button(window,text='Activate Timer',bg='Red', command=timer).pack(pady=40)

#Close the window and exit the app

window.mainloop()

**Code explanation:**

This snippet of code is optional

* **Button():** When the user selects or clicks on a button, a function is called. Thus to call functions or a specific task based on user input (interaction), we make use of buttons.

#### **Project output:**

Enter the time for the timer and view the notification:



#### **Summary**

We created a simple Countdown Clock and Timer with plyer, tkinter and time modules. The project explores many features of Tkinter such as entry widgets, mouse clicks, buttons and placeholders. Using two ways, notifications were created and used.