UET KSK

2020

Lab Report 1

(Introduction to computing)

Name:

RIFFAT JABEEN

Reg. No. :

2020-EE-291

Section:

B

Instructor:

Dr. Farrukh

Date of Submission: 16-jan-2021

Designing a GUI based Clock on Tkinter

Objective

This project aimed to design a Tkinter based GUI of clock in addition of counter clock to have clear knowledge of designing of graphic user interface on python and use of various objects and methods of tkinter library.

Introduction

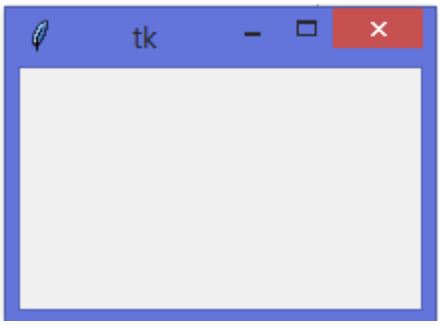
Upto now what we have done was just coding and understanding the Basic syntax of python language i.e. providing input at console and get output there. Know it is time to build something which can be brought to market for the user to interact. For this purpose python has specific libraries that contain pre-defined classes and methods (Object oriented programming) like Kivy, Libavg, PyQT, Pyforms etc. etc. But the most common and one which is widely used all over the world is Tkinter (pronounced as kinter).

Basic use of tkinter includes two function just to show up a GUI on desktop that are:

```
from tkinter import *

root = Tk()

root.mainloop()
```

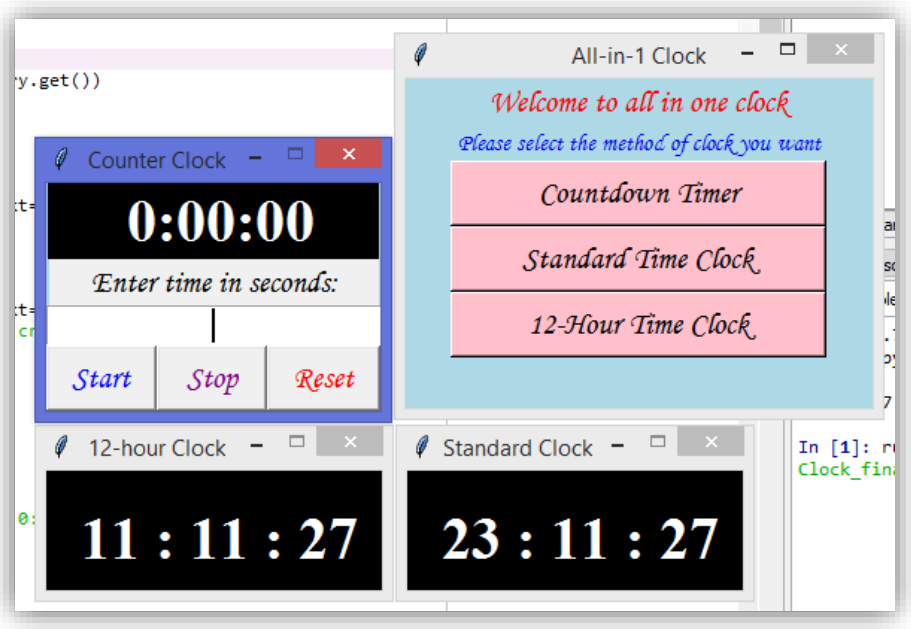


These three commands generate a GUI as shown in adjacent figure.

Design Methodology

Since I wanted to implement three features in single program so I opted the method of GUI appearing from one parent GUI. One Top level GUI containing the options in form of buttons and then each button has the command of a function which generates a new respective GUI.

Following figure will make the idea clear…



Explanation of code

To make a clear picture of how the whole precdure goes, lemme explain each part of code. Code starts with importing all needed libraries that will be required in upcoming code:

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jan 14 13:20:53 2021

@author: Riffat
"""
import tkinter as tk
```

importing any liberay as say xx means that in future we will be required to use xx instead of full name of liberary.

```
from tkinter import *
```

*Importing * from any library means that we are importing all functions of that library. I intentionally used this line to show that how we can use a function of library linked with that library "b1 = tkinter.Button()" or independently use all functions of library "b1 = Button()"*

```
from time import sleep
```

From time library I imported sleep function which will be used in counter clock to provide delay of 1 second.

```
import datetime
```

datetime library allows to fetch system's time and to tackle it with some other functions. I used it to fetch time, convert it to str, formatting it and to convert seconds in time for counter clock.

```
import winsound as ws
```

winsound library allows to use windows inbuilt sounds in our program. I used it as an alarm at the end of counter clock.

Making top level GUI:

`root = tk.Tk()`	*Starting GUI.*
`root.title("All-in-1 Clock")`	*sets the title of GUI as "All-in1 Clock"*
`root.geometry('300x200')`	*setting width and hight of parent GUI.*
`root.config(bg='light blue')`	*configures the background color of GUI to be light blue*

```
lbl1 = tk.Label(root,text="Welcome to all in one clock",font=("Monotype Corsiva",16),
bg="light blue", fg="red")
```

Creating a label named lbl1 using label function into GUI named `'root'` *that would have text* `"Welcome to all in one clock"` *and font* `"Monotype/ Corsiva"` *with the hight of* `16,` *background color light blue and forground color red.*

```
btn1 = tk.Button(root, text = "Countdown Timer",command=oneclick1, bg='pink', width=22,\
font=("Monotype Corsiva",16))
btn2 = tk.Button(root, text = "Standard Time Clock",command=oneclick2, bg='pink', width=22,\
font=("Monotype Corsiva",16))
btn3 = tk.Button(root,text = "12-Hour Time Clock",command=oneclick3,bg='pink', width=22,\
font=("Monotype Corsiva",16))
```

These three commands are for creating three buttons, each button corresponds to functions oneclick1, oneclick2 and oneclick3 respectively. These three function each will generate a separate GUI for counter clock, standard clock and 12-hour clock respectively. I have set the width of all buttons to be of 22 charchters in order to maintain symmetry.

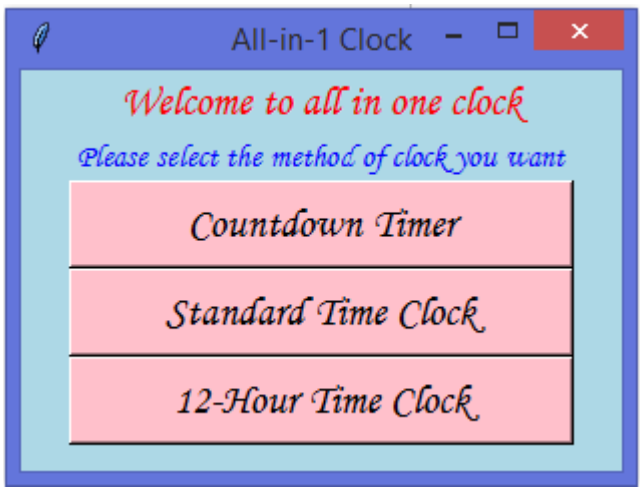


Figure 3: Screen capture of Top level GUI.

Function oneclick1 (counter clock GUI):

```
def oneclick1():
        counter = Tk()                          starting GUI

        counter.resizable(False, False)         making it non-resizeable both from hight and
width

        counter.title("Counter Clock")                  setting title as "Counter clock"

        counter.geometry('224x153')             setting height and width by pixels

        counter.config(bg='light blue')         setting light blue color as background color

        clock_label = Label(counter,text=" 0:00:00 ", bg="black",fg="white", font=\
        ("Times", 30, 'bold'), relief='flat')
```

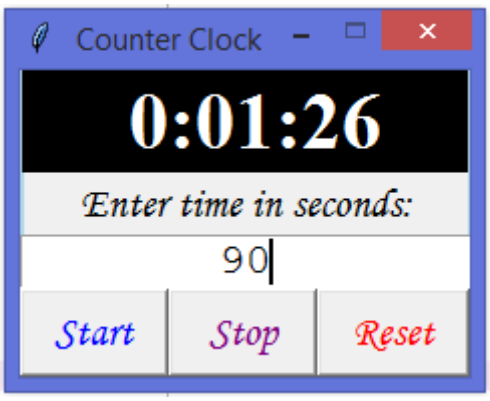
Creates a Label in counter GUI with text " 0:00:00 " of white color in the large bold font of 30pt of Times New Roman. This is the label where our counter will work.

```
        clock_label.grid(row=0,column=0)
```
placing this label in the full start of GUI.

```
        counterEntry = Entry(counter,width=17,font=("Courier",16),justify='center')
        counterEntry.grid(row=2,column=0)
```
placing that entry box below counter label.

Creates an Entry box to get input from user. User will be asked to enter the countdown time in seconds and the we will convert that seconds in time in upcoming commands.

```
cb1=Button(counter,text='Start,fg='blue',command=start,font=("Monotype\
Corsiva",16), width=5)
cb1.place(x=0,y=109,width=74)
cb2=Button(counter,text='Stop',fg='purple',command=stop,font=("Monotype\
Corsiva",16), width=5)
cb2.place(x=74,y=109,width=74)
cb3=Button(counter,text='Reset',fg='red', command=reset, font=("Monotype\
Corsiva",16), width=6)
cb3.place(x=148,y=109)
```



Below the entry, I made three buttons to start, stop and reset the counter clock. Font, width and foreground text color is set same like in labels and other tkinter objects. New thing here used is the palce function the function is irrespective of pack or grid. (as pack and grid can not be used simmultaniously) But it can be used. Pack function palces the tkinter objects with respect to mentioned pixels starting from top left corner. Since all three buttons were to be in same row, I used y dimension 109 pixels. And x equal to width of prevous button so that three button becom conjusted.

Function start (command of start button):

```
def start():
    global seconds_left
    seconds_left = int(counterEntry.get())
```

Here Creating a global variable '`seconds_left`*' and taking what user enters in entry box and depositing it in global veriale '*`seconds_left`*'. The reason of using global variable is that it is going to be used in reset function to stop execution of start function.*

```
    global loop = True      global variable used to run while loop and interrupt
                            through
        while loop:         the stop function.
```

Since `seconds_left` *was the number entered by the user so we have to run this loop that much times and each time with the delay of 1 second that number of seconds is converted into standard time (in from* `0:00:00`*) and then '*`clock_label`*' updated with the that time as text. And* `seconds_left` *variable is decremented by one since one second would have been passed by that time. See following commands:*

```
        if seconds_left>-1:
            clock_label.config(text=" "+\
            str(datetime.timedelta(seconds=seconds_left))+" ")
            clock_label.update()
            seconds_left-=1
```

This loop will run upto `seconds_left` *becomes 0. When it becomes 0 then following commands executed and the* `clock_label` *is configured to* `0:00:00`*. Here we use* `winsounds` *library to produce alarm sound as mentioned by following 3rd line and loop gets break.*

```
        if seconds_left==-1:
            clock_label.config(text="0:00:0
            0")
            ws.PlaySound("rooster crow ",\
            ws.SND_FILENAME)
            break
        sleep(1)
```

Function stop (command of stop button):

```
    def stop():         initiatizing function stop
      global loop       creating global variable loop, it will become the same
                        which was defined in  start function.
      loop = False      setting that global variable loop  False which was set
                        True as default.
```

It is clear that stop function is just making the global variable loop as False. Due to which loop gets terminated and the text which was configured at `counter_label` *will remain there.*

Function reset (command of reset button):

```
def reset():
    clock_label.config(text=" 0:00:00 ")
    counterEntry.delete(0,END)
    global seconds_left
    seconds_left=0
```

`clock_label.config(text=" 0:00:00 ")` — *configuring* `clock_label` *to default value i.e.* `0:00:00`

`counterEntry.delete(0,END)` — *it will clear whatever is written in entry box*

`global seconds_left` — *creating global variable* `seconds_left` *which will linked with that defined in* **`start`** *function.*

`seconds_left=0`

setting seconds_left to 0 so that first if condition of while loop of start function could get false and GUI go to termination with alarm if executing otherwise it would simply reset every object of counter GUI to its default states.

Function oneclick2 (Standard Clock GUI):

```
def oneclick2():
```

`from time import strftime` — *Importing strftime function from time library to get standard sytem time*

`from tkinter import Label, Tk` — *Importing Label and Tk function from tkinter liberary*

`window = Tk()` — *Creating GUI named window for standard clock time display*

`window.title("Standard Clock")` — *Setting windows tittle to be "Standard clock"*

`window.geometry("225x80")` — *this sets size of window in pixels.*

`window.configure(bg="black")` — *Set black background color*

`window.resizable(False, False)` — *Making window non-resizable.*

```
clock_label = Label(counter,text=" 0:00:00 ", bg="black",fg="white", font=\
("Times", 30, 'bold'), relief='flat')
```

Since, whole GUI is a single Label of standard time, this one command is making the whole format of GUI. In which we are setting default text to be 0:00:00 of large font of 30pt Times New Roman in white color and background is black.

```
def update_label():
    current_time = strftime('%H : %M : %S')
```

`strftime` *function is the main worker of this GUI which is taking system's time and making a string in the* `hours: minutes: seconds` *formate, this str we can easily send to configure label.*

```
clock_label.configure(text = current_time)
```

configruring label by the string got by above command.

```
clock_label.after(80, update_label)
```

updates the label after each 80 milliseconds

`update_label()` *implementing above function*
`window.mainloop()` *making GUI ending.*

Figure 5: Screen capture of standard clock

Function oneclick3 (12-Hour Clock GUI):

In the previous (Standard Clock) GUI and this (12-Hours Clock GUI there is a single difference and that is in implimenting **`strftime`** *function.*
Here it is implimented as:

```
current_time = strftime('%I : %M : %S')
```

there %H was formating it in form of 24 hours format and here %I is formating it as 12 hours format. Then in the same way this str is send to configure the Label and updated after each 80 miliseconds.

Figure 6: Screen capture of 12-hour clock

Conclusion:

Conclusively It can be said that in this program that I built, I tried to use maximum functions and formating styles So that the person reading this report (anyone) can get macimum knowledge of building GUIs in python. Creating windows from windows, using various fonts and colors, using liberaries, and designing a charming GUI so that one interacting with is can never get bored.

THE END