```
from datetime import timedelta,datetime
from math import sin, cos, pi
from tkinter import *

day_list    = ['MONDAY','TUESDAY','WEDNESDAY','THURSDAY','FRIDAY','SATURDAY','SUNDAY']
month_list  = ['January','February','March','April','May','June','July','August','September','October','November','December']

class find_coords:
    def __init__(self,world,viewport):
        x_min,y_min,x_max,y_max     = world     #the coordinates we want to convert it to
        X_min,Y_min,X_max,Y_max     = viewport  #the area we want to convert to the new coordinates
        self.delta                  = min((X_max-X_min)/(x_max-x_min),(Y_max-Y_min)/(y_max-y_min))  #the scale / 1 unit corresponds to delta value
        x_c,y_c                     = 0.5*(x_min+x_max),0.5*(y_min+y_max)   #the center shift required from the centerr of the screen
        X_c,Y_c                     = 0.5*(X_min+X_max),0.5*(Y_min+Y_max)   #the center of the screen
        self.c_1,self.c_2           = X_c-self.delta*x_c,Y_c-self.delta*y_c #the center we want(the clock)
    def change(self,x,y):
        X_new,Y_new                 = (self.delta*x+self.c_1),(self.delta*(-y)+self.c_2)    #defining the new cooords. of the point x,y
        return X_new,Y_new                                  #return the new coords.
    def coords(self,x1,y1,x2,y2):
        return self.change(x1,y1),self.change(x2,y2)    #calls change function twice to convert a pair of coordinates

class make_clock:
    def __init__(self,root,time_gap=0,w=1000,h=650):
        self.def_world              = [-1,-1,1,1.5]     #the coordinates we want to convert the reference area to
        width,height                = w,h               #the width and the height of the widget
        self.refresh_time           = 500               #the intervals of time at which the clock must be refreshed
        self.time_gap               = timedelta(hours = time_gap)   #returns the diff. of the datetime values inthe datetime form (we use it here to convert the lag to datetime format)
        self.gap                    = min(width,height)/16          #the minimum gap to be left from the edges of the widget

        self.color_bg               = 'black'           #the background color
        self.color_clock_details    = 'white'           #the color of the hour numbers and of the digital time
        self.color_date_at_month    = 'red'             #the color of the date written at the hour no. equal to the month
        self.color_main_needles     = 'light blue'      #the color of the needles of the main clock
        self.color_inner_circles    = 'blue'            #the color of the numbers of the inner dials and the pin at the center
        self.color_inner_needles    = 'red'             #the color of the needles of the inner dials

        self.font_main_details      = 'Harrington 20'   #the font face and size of the hour nos., date and the digital clock
        self.font_inner_titles      = 'Papyrus 10'      #the font face and size of the titles of the inner dials and the day

        self.center_circle_rad      = 0.05/2            #the radius of the pin at the center
        self.len_hrs                = 0.50              #the length of the hour-hand
        self.len_min                = 0.80              #the length of the minute-hand
        self.len_sec                = 0.90              #the length of the second-hand
        self.width_hrs          = (self.gap-10)/3       #the width of the hour hand
        self.width_min          = (self.gap-10)/6       #the width of the minute hand

```

```
47          self.inner_cl_no_font       = {'Month':'calibri 9','Day':'calibri
            9','Date':'calibri 7 bold'}   #the font face, size and style(optionally) of
            the month, day and date dials
48          self.inner_cl_radius        =
            {'Month':0.2,'Day':0.2,'Date':0.35}                                 #the
            radius of the month, day and date dials
49          self.inner_cl_loc           =
            {'Month':(0,0.4),'Day':(-0.5,0),'Date':(0,-0.5)}                    #the
            location of the centers of the month, day and date dials
50          self.inner_cl_count         =
            {'Month':12,'Day':7,'Date':31}                                      #the no of
            ticks/digits of the month, day and date dials
51          self.inner_cl_width_needles =
            {'Month':4,'Day':4,'Date':4}                                        #the width
            of the needles of the month, day and date dials
52          self.inner_cl_detail        = {'Month':'self.month','Day':'self.w_day
            +1','Date':'self.date'}   #the values of the month, day and date dials
53
54          viewport                    =
            (self.gap,self.gap,width-self.gap,height-self.gap)                  #the area
            we want to work with
55          self.convert                = find_coords(self.def_world,viewport)
56
57          self.root                   = root
58          self.canvas                 =
            Canvas(root,width=width,height=height,background=self.color_bg)
59          self.canvas.pack(fill=BOTH,expand=True)
60          self.canvas.bind("<Configure>",self.resize)
61          self.action()
62
63      def resize(self,event):
64          self.canvas.delete(ALL)
65          width,height                =
            self.canvas.winfo_width(),self.canvas.winfo_height()
66          self.gap                    = min(width,height)/16
67          viewport                    =
            (self.gap,self.gap,width-self.gap,height-self.gap)
68          self.convert                = find_coords(self.def_world,viewport)
69
70      def action(self):
71          self.create()
72          self.root.after(self.refresh_time,self.action)
73
74      def create(self):
75          self.canvas.delete(ALL)
76          self.year,self.month,self.date,self.hrs,self.min,self.sec,self.w_day,*extra
            = datetime.timetuple(datetime.utcnow()-self.time_gap)
77          self.month_name             = month_list[self.month-1]
78          start,step                  = pi/2,pi/6
79          for i in range(12):
80              angle                   = start-(i*step)
81              x_coord,y_coord         = cos(angle),sin(angle)
82              if i==self.month or (i+12)==self.month:
83
                    self.canvas.create_text(self.convert.change(x_coord,y_coord),fill=self
                    .color_date_at_month,text='
                    '.join([str(self.date),self.month_name[:3],str(self.year)]),font=self.
                    font_main_details)
84              elif i==0:
85
                    self.canvas.create_text(self.convert.change(x_coord,y_coord),fill=self
                    .color_clock_details,text=str(12),font=self.font_main_details)

86              else:
87
                    self.canvas.create_text(self.convert.change(x_coord,y_coord),fill=self
                    .color_clock_details,text=str(i),font=self.font_main_details)
88          self.Create_Dial('Month')
89          self.Create_Dial('Day')
90          self.Create_Dial('Date')
91          self.create_needles()
92          self.create_digital()
```

```
93
94
95      def Create_Dial(self,name):
96          start,step                  = pi/2,2*pi/self.inner_cl_count[name]
97          for i in range(self.inner_cl_count[name]):
98              angle                   = start-(i*step)
99              x_coord,y_coord         =
                self.inner_cl_radius[name]*cos(angle),self.inner_cl_radius[name]*sin(angle
                )
100
                self.canvas.create_text(self.convert.change(x_coord+self.inner_cl_loc[name
                ][0],y_coord+self.inner_cl_loc[name][1]),fill=self.color_inner_circles,tex
                t=str(i+1),font=self.inner_cl_no_font[name])
101         angle                       =
            start-((eval(self.inner_cl_detail[name])-1)*step)
102         x_month,y_month             =
            (self.inner_cl_radius[name]-0.05)*cos(angle),(self.inner_cl_radius[name]-0.05)
            *sin(angle)
103
            self.canvas.create_line(self.convert.coords(self.inner_cl_loc[name][0],self.in
            ner_cl_loc[name][1],x_month+self.inner_cl_loc[name][0],y_month+self.inner_cl_l
            oc[name][1]),fill=self.color_inner_needles,width=self.inner_cl_width_needles[n
            ame])
104
            self.canvas.create_text(self.convert.change(self.inner_cl_loc[name][0],self.in
            ner_cl_loc[name][1]+0.05),fill=self.color_clock_details,text=name,font=self.fo
            nt_inner_titles)
105
106
107     def create_needles(self):
108         angle                       = (pi/2)-(pi/6)*(self.hrs+(self.min/60.0))
109         x_hrs,y_hrs                 = cos(angle)*self.len_hrs,sin(angle)*self.len_hrs
110         angle                       = (pi/2)-(pi/30)*(self.min+(self.sec/60.0))
111         x_min,y_min                 = cos(angle)*self.len_min,sin(angle)*self.len_min
112         angle                       = (pi/2)-(pi/30)*self.sec
113         x_sec,y_sec                 = cos(angle)*self.len_sec,sin(angle)*self.len_sec
114         draw_line                   = self.canvas.create_line
115
            draw_line(self.convert.coords(0,0,x_hrs,y_hrs),fill=self.color_main_needles,wi
            dth=self.width_hrs)
116
            draw_line(self.convert.coords(0,0,x_min,y_min),fill=self.color_main_needles,wi
            dth=self.width_min)
117
            draw_line(self.convert.coords(0,0,x_sec,y_sec),fill=self.color_main_needles,ar
            row='last')
118
            self.canvas.create_oval(self.convert.coords(-self.center_circle_rad,-self.cent
            er_circle_rad,self.center_circle_rad,self.center_circle_rad),fill=self.color_i
            nner_circles)
119
120
121     def create_digital(self):
122         day_name=day_list[self.w_day]
123
            self.canvas.create_text(self.convert.change(0.5,0),fill=self.color_clock_detai
            ls,text=day_name,font=self.font_inner_titles)
124         time_display                = '%02i : %02i :
            %02i'%(self.hrs,self.min,self.sec)
125         date_display                = '
            '.join([str(self.date),self.month_name,str(self.year)])
126         self.root.title(time_display)
127
            self.canvas.create_text(self.convert.change(0,-1.3),fill=self.color_clock_deta
            ils,text=time_display,font=self.font_main_details)
128
            self.canvas.create_text(self.convert.change(0,-1.45),fill=self.color_clock_det
            ails,text=date_display,font=self.font_main_details)
129 time_gap=input("\n\nEnter the difference between utc and your time zone\n(negetive
    meaning time zone is ahead),\nif not known press '?'\n")
130 try:
131     if time_gap=="?" or eval(time_gap):
```

```
        pass
except:
    time_gap=input("\n\nEnter the difference between utc and your time
    zone\n(negetive meaning time zone is ahead),\nif not known press '?'\n")
if time_gap=='?':
    print ('\n\nPlease choose a country...')
    print ('Press 1 for India')
    print ('Press 2 for Australia')
    print ('Press 3 for USA')
    i=int(input())
    while not(i==1 or i==2 or i==3):
        print ('Please choose a valid option...')
        i=int(input())
    if i==1:
        time_gap=-5.5
    elif i==2:
        time_gap=-11
    elif i==3:
        time_gap=5
else:
    time_gap=float(time_gap)

w,h         = 500,650
root        = Tk()
make_clock(root,time_gap,w,h)
root.mainloop()
```