A Minor Project in Python

Project Title: Creation of A Countdown Timer Using Python

Domain: Data Science

Submitted

To

Teachnook

Submitted

By

Name: ANIKA MARIAM

Student of 1st Semester

Branch: Computer Science Engineering

Kallinga Institute of Industrial Technology, Bhubaneswar

ABSTRACT

The project entitled “Creation of A Countdown Timer Using Python” has been developed by using various features of Python. It has been developed by creating a class that contains different functions for including different features like ***Start, Stop, Reset, Pause, Resume*** and ***Exit.*** These features have been implemented using different buttons. A Single button for ***Start*** and ***Stop*** features has been used i.e. once the ***Start*** button is clicked then it becomes ***Stop*** button and vice-versa. In the same way, a Single button for ***Pause*** and ***Resume*** features has been used i.e. once the ***Pause*** button is clicked then it becomes ***Resume*** button and vice-versa .

Coding for the design of Countdown Timer

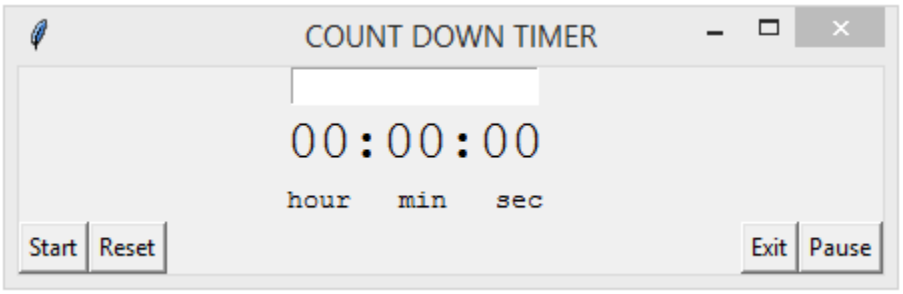
```
import tkinter as tk
import tkinter.messagebox
import time

class Count_Down_Timer(tk.Frame):
    def __init__(self, master, *args, **kwargs):
        tk.Frame.__init__(self, master, *args, **kwargs)
        self.master = master
        self.running = False
        self.time = 0
        self.hours = 0
        self.mins = 0
        self.secs = 0
        self.display_timer()

    def display_timer(self):
        self.time_entry = tk.Entry(self)
        self.time_entry.grid(row=1, column=1)

        self.clock = tk.Label(self, text="00:00:00", font=("Courier", 20), width=20)
        self.clock.grid(row=2, column=1, stick="S")

        self.time_label = tk.Label(self, text="hour   min   sec", font=("Courier", 10), width=25)
        self.time_label.grid(row=3, column=1, sticky="N")

        self.power_button = tk.Button(self, text="Start", command=lambda: self.start())
        self.power_button.grid(row=4, column=0, sticky="NE")

        self.reset_button = tk.Button(self, text="Reset", command=lambda: self.reset())
        self.reset_button.grid(row=4, column=1, sticky="NW")

        self.exit_button = tk.Button(self, text="Exit", command=lambda: self.exit())
        self.exit_button.grid(row=4, column=2, sticky="NE")

        self.pause_button = tk.Button(self, text="Pause", command=lambda: self.pause())
        self.pause_button.grid(row = 4,column=3, sticky = "NW")

        self.master.bind("<Return>", lambda x: self.start())
```

```
    self.time_entry.bind("<Key>", lambda v: self.update())

def calculate(self):
    self.hours = self.time // 3600
    self.mins = (self.time // 60) % 60
    self.secs = self.time % 60
    return "{:02d}:{:02d}:{:02d}".format(self.hours, self.mins, self.secs)

def update(self):
    self.time = int(self.time_entry.get())
    try:
        self.clock.configure(text=self.calculate())
    except:
        self.clock.configure(text="00:00:00")

def timer(self):
    if self.running:
        if self.time <= 0:
            self.clock.configure(text="Time is over!")
        else:
            self.clock.configure(text=self.calculate())
            self.time -= 1
            self.after(1000, self.timer)

def start(self):
    try:
        self.time = int(self.time_entry.get())
        self.time_entry.delete(0, 'end')
    except:
        self.time = self.time
    self.power_button.configure(text="Stop", command=lambda: self.stop())
    self.master.bind("<Return>", lambda x: self.stop())
    self.running = True
    self.timer()

def reset(self):
    self.power_button.configure(text="Start", command=lambda: self.start())
    self.master.bind("<Return>", lambda x: self.start())
    self.running = False
    self.time = 0
```

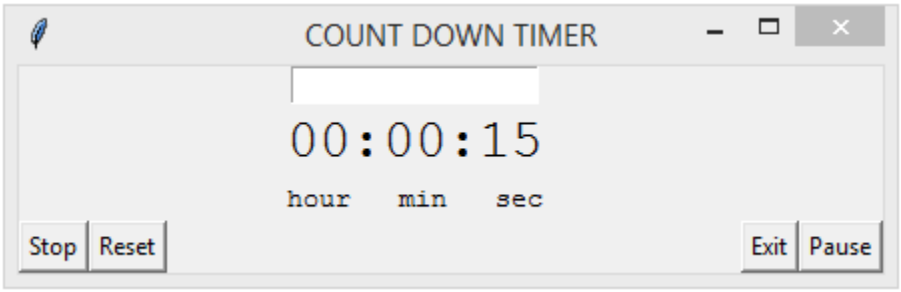
```
        self.clock["text"] = "00:00:00"
    def stop(self):
        self.clock.configure(text="Stopped Externally")
        self.power_button.configure(text="Start", command=lambda: self.start())
        self.master.bind("<Return>", lambda x: self.start())
        self.running = False
        self.time = 0
    def pause(self):
        self.pause_button.configure(text="Resume", command=lambda: self.resume())
        self.master.bind("<Return>", lambda x: self.resume())
        if self.running == True:
            self.running = False
        self.timer()


    def resume(self):
        self.pause_button.configure(text="Pause", command=lambda: self.pause())
        self.master.bind("<Return>", lambda x: self.pause())
        if self.running == False:
            self.running = True
        self.timer()
    def exit(self):
        self.running = False
        if tk.messagebox.askokcancel("Exit", "Click on OK if you really want to exit?"):
            root.destroy()
        else:
            self.running = True
            self.timer()


if __name__ == "__main__":
    root = tk.Tk()
    root.title("COUNT DOWN TIMER")
    Count_Down_Timer(root).pack(side="top", fill="both", expand=True)
    root.mainloop()
```

Different Outputs

Output-1

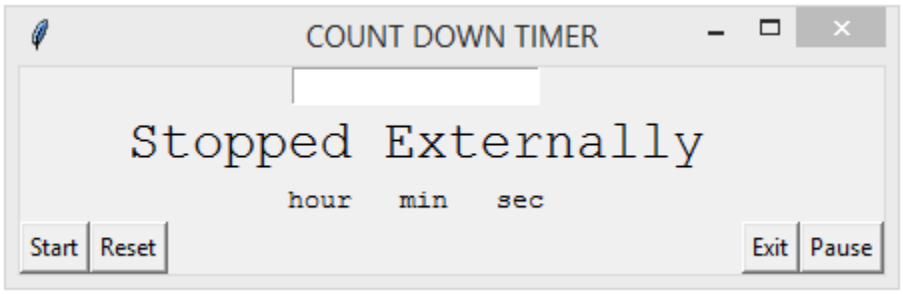


Output-2

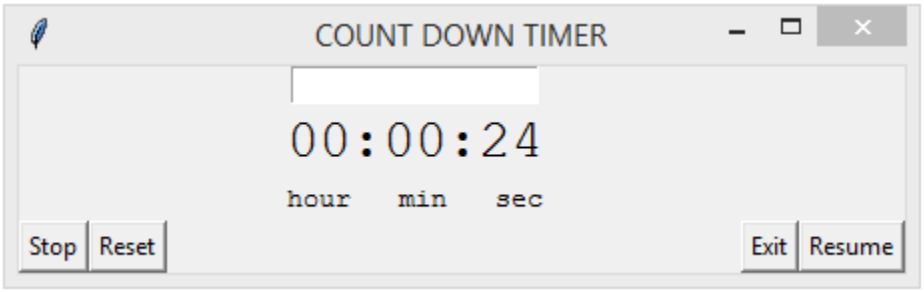


Output-3

Output-4



Output-5



Output-6