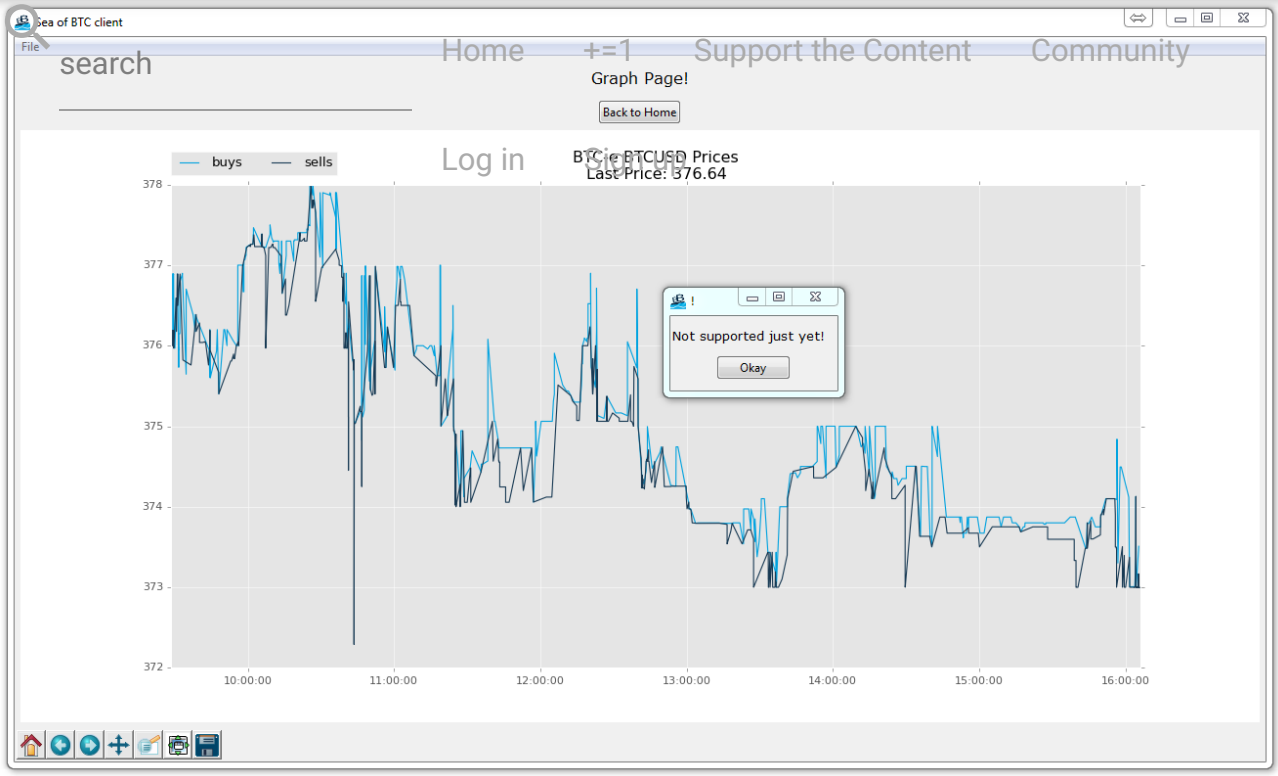
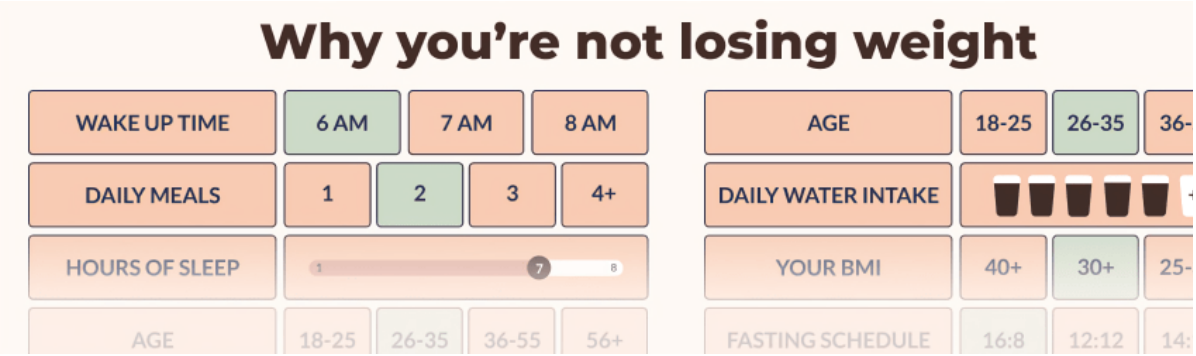


Building a pop-up message window

Pop-up message window - Tkinter tutorial Python 3.4 part 12

Many times, it is generally useful to have a quick and easy way to communicate with your user. There are many ways this can be done, with some text at the top of the window, or with something like a pop up message. Text at the top of the window is very easy to miss, so most people choose to do a pop-up message. This is used for things like informational messages, warnings, disclaimers, new product versions, and a whole lot more.

The goal is to create something that we can use in a variety of circumstances, so we want a popup window function, along with a text parameter where we can specify the text we want to show.

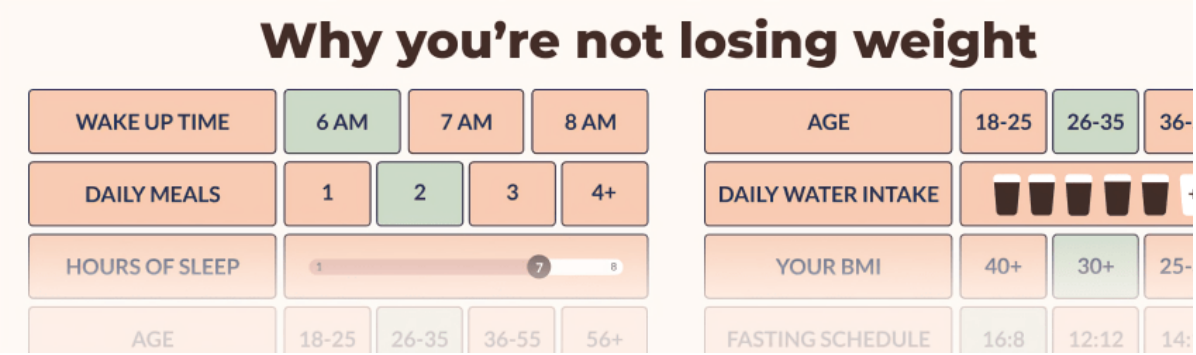


To start, we're going to need a couple new fonts, since "large_font" is a bit big.


```
LARGE_FONT= ("Verdana", 12)
NORM_FONT = ("Helvetica", 10)
SMALL_FONT = ("Helvetica", 8)
```

Now that we have that, we're going to create our popup function:

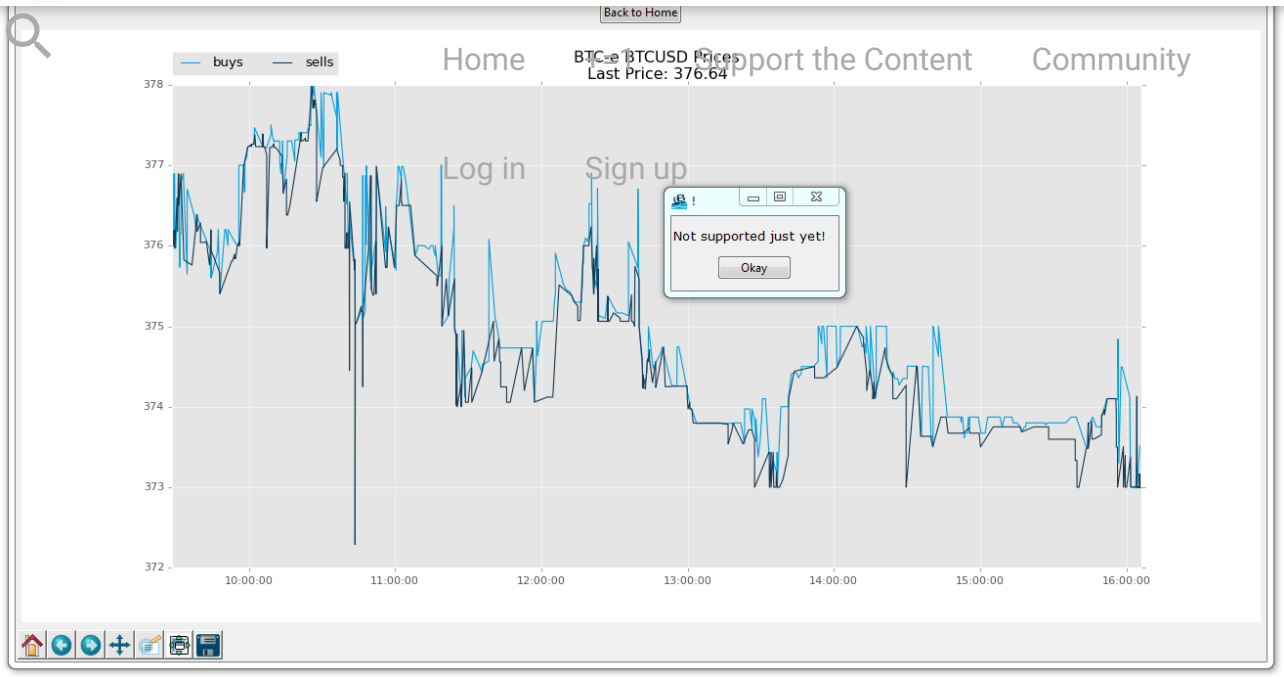
```
popup.wm_title("!")
label = ttk.Label(popup, text=msg, font=NORM_FONT)
label.pack(side="top", fill="x", pady=10)
B1 = ttk.Button(popup, text="Okay", command = popup.destroy)
B1.pack()
popup.mainloop()
```



Here, we're defining a new tk.Tk instance, giving it a new title ("!"), and then preparing our text label with the msg parameter from the function. From there, we create a simple "okay" button, which will destroy the window. We pack that, and we're all set!

That's it, now when we want to have a popup message, say with a button, we would just have something like: command = lambda: popupmsg("popup message here!"))

The result now if you go to File and choose save settings:



The full code:

```
# The code for changing pages was derived from: http://stackoverflow.com/qu
# License: http://creativecommons.org/licenses/by-sa/3.0/

import matplotlib
matplotlib.use("TkAgg")
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg, Navigation
from matplotlib.figure import Figure
import matplotlib.animation as animation
from matplotlib import style

import tkinter as tk
from tkinter import ttk

import urllib
import json

import pandas as pd
import numpy as np

from matplotlib import pyplot as plt



LARGE_FONT= ("Verdana", 12)
NORM_FONT= ("Verdana", 10)
SMALL_FONT= ("Verdana", 8)
```

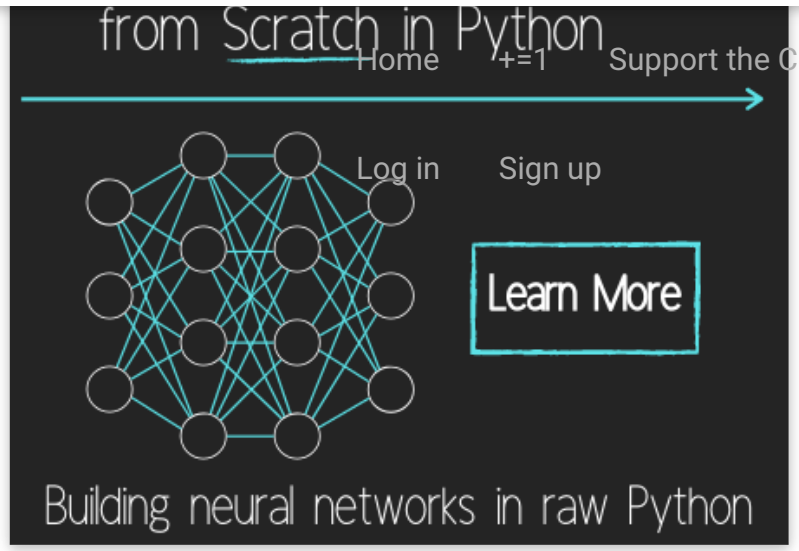
```
style.use("ggplot")

f = Figure()
a = f.add_subplot(111)


def popupmsg(msg):
    popup = tk.Tk()
    popup.wm_title("!")
    label = ttk.Label(popup, text=msg, font=NORM_FONT)
    label.pack(side="top", fill="x", pady=10)
    B1 = ttk.Button(popup, text="Okay", command = popup.destroy)
    B1.pack()
    popup.mainloop()



def animate(i):
    dataLink = 'https://btc-e.com/api/3/trades/btc_usd?limit=2000'
    data = urllib.request.urlopen(dataLink)
    data = data.readall().decode("utf-8")
    data = json.loads(data)


    data = data["btc_usd"]
    data = pd.DataFrame(data)

    buys = data[(data['type']=="bid")]
    buys["datestamp"] = np.array(buys["timestamp"]).astype("datetime64[s]")
    buyDates = (buys["datestamp"]).tolist()


    sells = data[(data['type']=="ask")]
    sells["datestamp"] = np.array(sells["timestamp"]).astype("datetime64[s]
    sellDates = (sells["datestamp"]).tolist()

    a.clear()

    a.plot_date(buyDates, buys["price"], "#00A3E0", label="buys")
    a.plot_date(sellDates, sells["price"], "#183A54", label="sells")

    a.legend(bbox_to_anchor=(0, 1.02, 1, .102), loc=3,
```


```
    title = "BTC-e BTCUSD Prices\nLast Price: "+str(data["price"][1999])
    a.set_title(title)




class SeaofBTCapp(tk.Tk):

    def __init__(self, *args, **kwargs):

        tk.Tk.__init__(self, *args, **kwargs)

        tk.Tk.iconbitmap(self, default="clienticon.ico")
        tk.Tk.wm_title(self, "Sea of BTC client")


        container = tk.Frame(self)
        container.pack(side="top", fill="both", expand = True)
        container.grid_rowconfigure(0, weight=1)
        container.grid_columnconfigure(0, weight=1)


        menubar = tk.Menu(container)
        filemenu = tk.Menu(menubar, tearoff=0)
        filemenu.add_command(label="Save settings", command = lambda: popup
        filemenu.add_separator()
        filemenu.add_command(label="Exit", command=quit)
        menubar.add_cascade(label="File", menu=filemenu)

        tk.Tk.config(self, menu=menubar)

        self.frames = {}

        for F in (StartPage, BTCe_Page):

            frame = F(container, self)
```


```
            frame.grid(row=0, column=0, sticky="nsew")
        self.show_frame(StartPage)

    def show_frame(self, cont):

        frame = self.frames[cont]
        frame.tkraise()


class StartPage(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self,parent)
        label = tk.Label(self, text=("""ALPHA Bitcoin trading application
        use at your own risk. There is no promise
        of warranty."""), font=LARGE_FONT)
        label.pack(pady=10,padx=10)

        button1 = ttk.Button(self, text="Agree",
                             command=lambda: controller.show_frame(BTCe_Page
        button1.pack()

        button2 = ttk.Button(self, text="Disagree",
                             command=quit)
        button2.pack()



class PageOne(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        label = tk.Label(self, text="Page One!!!", font=LARGE_FONT)
        label.pack(pady=10,padx=10)

        button1 = ttk.Button(self, text="Back to Home",
                             command=lambda: controller.show_frame(StartPage
        button1.pack()
```

```
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        label = tk.Label(self, text="Graph Page!", font=LARGE_FONT)
        label.pack(pady=10,padx=10)

        button1 = ttk.Button(self, text="Back to Home",
                             command=lambda: controller.show_frame(StartPage
        button1.pack()




        canvas = FigureCanvasTkAgg(f, self)
        canvas.show()
        canvas.get_tk_widget().pack(side=tk.BOTTOM, fill=tk.BOTH, expand=Tr

        toolbar = NavigationToolbar2TkAgg(canvas, self)
        toolbar.update()
        canvas._tkcanvas.pack(side=tk.TOP, fill=tk.BOTH, expand=True)



app = SeaofBTCapp()
app.geometry("1280x720")
ani = animation.FuncAnimation(f, animate, interval=5000)
app.mainloop()
```

The next tutorial: Exchange Choice Option



- Programming GUIs and windows with Tkinter and Python Introduction
- Object Oriented Programming Crash Course with Tkinter
- Passing functions with Parameters in Tkinter using Lambda
- How to change and show a new window in Tkinter
- Styling your GUI a bit using TTK
- How to embed a Matplotlib graph to your Tkinter GUI
- How to make the Matplotlib graph live in your application
- Organizing our GUI
- Plotting Live Updating Data in Matplotlib and our Tkinter GUI
- Customizing an embedded Matplotlib Graph in Tkinter
- Creating our Main Menu in Tkinter

Building a pop-up message window

- Exchange Choice Option
- Time-frame and sample size option

Trading option, start/stop, and help menu options

Tutorial on adding a tutorial

Allowing the exchange choice option to affect actual shown exchange

Adding exchange choice cont'd

Adding exchange choices part 3

Indicator Support

Pulling data from the Sea of BTC API

Setting up sub plots within our Tkinter GUI

Graphing an OHLC candlestick graph embedded in our Tkinter GUI

Acquiring RSI data from Sea of BTC API

Acquiring MACD data from Sea of BTC API

Converting Tkinter application to .exe and installer with cx_Freeze

You've reached the end!

Contact: Harrison@pythonprogramming.net.

Programming is a superpower.