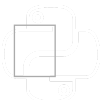


Plotting Live Updati Data in Matplotlib and our Tkinter GUI

Plotting live bitcoin price data - Tkinter GUI development series p. 9





Home

+=1

Support the Content

Community

```
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"])
    a.plot_date(sellDates, sells["price"])
```

Above, we're pulling the data from the BTC-e public API, then using the json module to actually process the data. From there, we're dividing the data into "bid" and "ask" data.



others.

[Home](#)

[+=1](#)

[Support the Content](#)

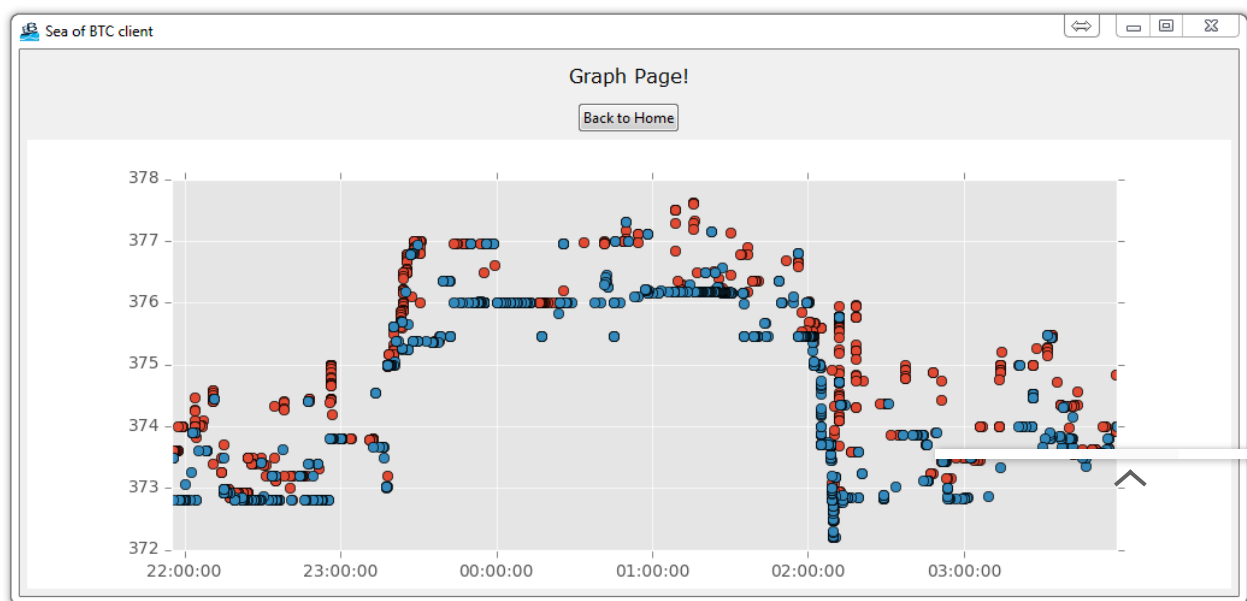
[Community](#)

From here, we convert the date data to datetime64, and then we're ready to graph.

For more information on that, see the video [Sign up](#)

That's it, though in the video we also chop out the starting button page, and we're now just instantly loading into the BTC-e page.

The finished product up to here when viewing the graph page should look like:



In case you're lost or confused, here's the full code:

```
# The code for changing pages was derived from: http://stackoverflow.com/qa
# 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
```



LARGE_FONT= ("Verdana", 12)
style.use("ggplot")

Home +=1

Support the Content

Community

Log in Sign up

f = Figure(figsize=(10,6), dpi=100)

a = f.add_subplot(111)

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"])

a.plot_date(sellDates, sells["price"])

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)

```

```

self.frames = {}

```

```

for F in (StartPage, BTCE_Page):

```

```

    frame = F(container, self)

```

```

    self.frames[F] = frame

```

```

    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()

```



```

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()

class BTcE_Page(tk.Frame):

    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=True)

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

app = SeaofBTCapp()
ani = animation.FuncAnimation(f, animate, interval=1000)
app.mainloop()

```



Home +1 Support the Content Community

software for your digital enterprise

Log in Sign up

Try ServiceDesk Plus!

ManageEngine
ServiceDesk Plus

The next tutorial: Customizing An Embedded Matplotlib Graph In Tkinter

Neural Networks from Scratch in Python

Learn More

Building neural networks in raw Python

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 make the Matplotlib graph live in your application

Home +=1 Support the Content Community

Organizing our GUI

Log in Sign up

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
- Adding indicator Menus (3 videos)
- 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



Home +=1 Support the Content Community



Log in Sign up

Visit The adidas® Store

You've reached the end!

Contact: Harrison@pythonprogramming.net.

Support this Website!
Consulting and Contracting
Facebook
Twitter
Instagram

Legal stuff:
Terms and Conditions
Privacy Policy



© OVER 9000! PythonProgramming.net

Programming is a superpower.