

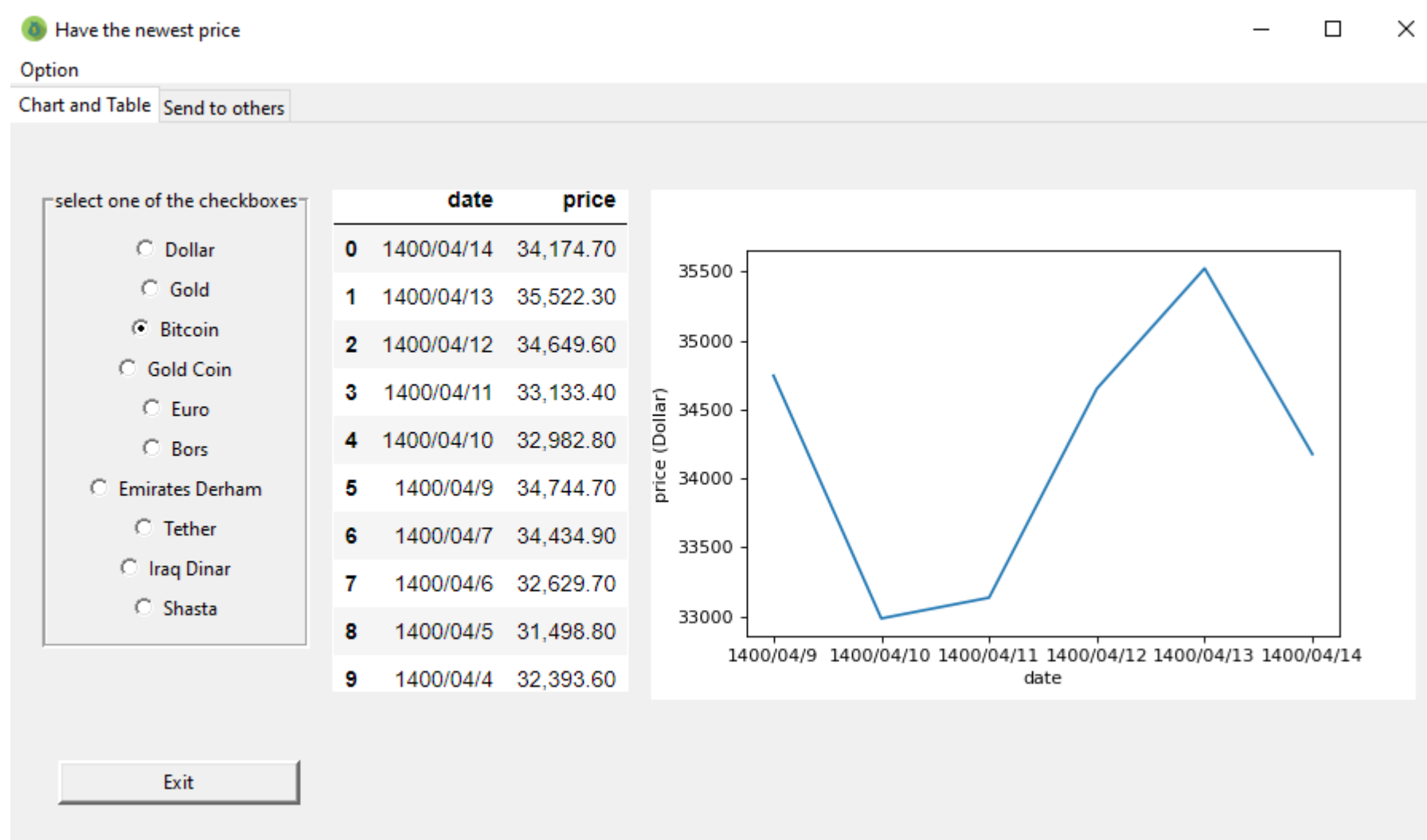
**مقدمه :**

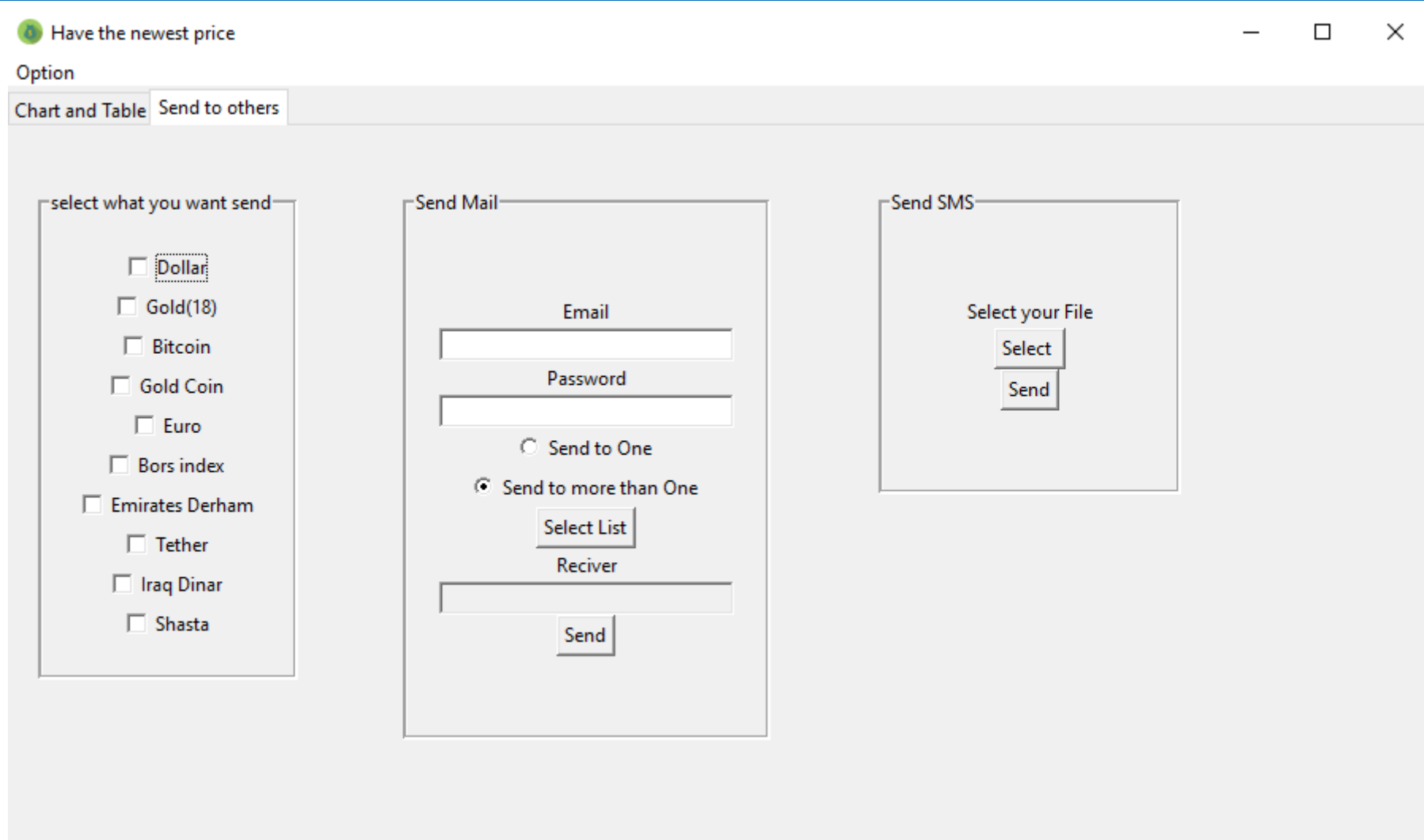
از آنجایی که حوزه سرمایه گذاری در زمینه های مختلف در سال های اخیر از اهمیت زیادی برخوردار شده تصمیم گرفتیم برنامه ای تو این زمینه بنویسم در طول زمان به این ایده رسیدیم که برنامه ای باشد که قیمت به روز ارز دلار را به ما در صفحه گرافیکی نمایش دهد که آرام آرام برخی دیگر از موارد مانند قیمت رمز ارز بیتکوین و ... را نیز اضافه کردم.

برنامه از دو بخش اصلی تشکیل شده بخش اول که پس از اجرا شدن ظاهر میشود بخش اصلی برنامه است که ۱۰ تا چک باکس را نمایش میدهد با کلیک بر روی هر کدام جدول و نمودار قیمت آن را نمایش میدهد جدول شامل تاریخ و قیمت آن در آن روز و در نمودار نیز تغییرات ۶ روز قبل را نمایش میدهد.

در بخش دوم امکان ارسال آخرین قیمت از طریق هم sms و هم mail این کار را انجام دهیم . که باز ارسال ایمیل را به دو بخش تقسیم کردم بخش ارسال به یک نفر و بخش ارسال به چند نفر که در بخش ارسال به دو نفر میبایست یک فایل txt که شامل ایمیل ها هست را به برنامه مشخص کنیم.

در بخش ارسال از طریقه شماره نیز باید اینگونه کار را پیش برد که اول باید یک فایل txt را به برنامه معرفی کرد سپس پس از فشردن دکمه ارسال قیمت موارد انتخاب شده را به شماره های موجود در لیست خواهد فرستاد.

**بخش اول برنامه**



## بخش دوم برنامه

### کد برنامه:

کتابخانه های اصلی که استفاده کردم شامل :

BeautifulSoup که برای اخراج داده از تگ های html استفاده میشود

Requests که برای دریافت فایل html با استفاده از متد get

Tkinter برای تشکیل رابط گرافیکی

Pandas برای تشکیل و ذخیره dataframe ها

Matplotlib برای ساخت نمودار

Smtplib و ssl برای ارسال ایمیل

MIMEMultipart برای مرتب کردن و قرار دادن داده ها درست در جای خود برای ارسال ایمیل

Threading گاهی اوقات برای افزایش سرعت از خاصیت threading استفاده کردم تا کاربر زمان کمتر را انتظار بکشد

http.client برای ارسال درخواست به sms server تا اس ام اس مورد نظر را به کاربر ارسال کند

dfi برای ذخیره dataframe به صورت png

برنامه در کلاس price قرار دارد.

که کلاس price دارای یک تابع init و ۱۸ تابع دیگر است.

در تابع init ابتدا فایل های مربوط به tkinter را ساختیم تا برنامه بلافاصله پس از اجرا رابط گرافیکی ساخته و نمایش داده شود.

self.root = tkinter.Tk() از نوع self هم قرار دادم چون لازم میشود در توابع مختلف به آن باید دسترسی داشته باشیم.

بقیه ویژگی ها نیز بعد از تعریف self.root به آن اضافه کردیم مانند اندازه صفحه و رنگ و ...

در ادامه تابع init چو میخواستیم برنامه دو بخش داشته باشد از ویژگی tab در tkinter استفاده کردم.

```
tabs = ttk.Notebook(self.root)
self.first = ttk.Frame(tabs)
tabs.add(self.first, text='Chart and Table')
self.second = ttk.Frame(tabs)
tabs.add(self.second, text='Send to others')
```

در tab اول یا صفحه اول نمایش نمودار و جدول قیمت را داریم که برای زیبایی ده تا Radiobutton را درون یک LabelFrame قرار دادم که با زدن هر کدام نمودار و جدول قیمت در همان tab اول ظاهر میشود برای بررسی دقیق تر

```
Radiobutton(frame, text='dollar', variable=self.1, value='dollar', command=self.dollar_price).pack
```

با کلیک روی این باکس بلافاصله تابع self.dollar\_price اجرا میشود که در ادامه بیشتر به آن میپردازیم.

بخش بعدی تابع init مربوط به صفحه دوم (tab2) هست که ما باز در آن ۱۰ چک باکس داریم که البته باکس های ما از نوع Checkbutton هست در این بخش نیاز به IntVar نیاز داریم تا بفهمیم کاربر کدام باکس ها را کلیک کرده است تا در بخش ارسال مقدار های باکس هایی که ۱ هست را برای کاربران ارسال کنیم

به این شکل همانطور که مشاهده میکنید.

```
self.dollar = IntVar()
self.gold = IntVar()
self.bitcoin = IntVar()
self.coin = IntVar()
self.euro = IntVar()
self.bors = IntVar()
self.derham = IntVar()
self.tether = IntVar()
```

```
self.iraq = IntVar()  
self.shasta = IntVar()
```

که مقدار هر کدام در صورت تیک زدن کاربر یک میشود و وقتی که کاربر میخواهد این اطلاعات را بفرستد در صورتی که هر کدام از آن ها مقدار آن یک باشد مقدار آن دانلود و ارسال خواهد شد.

در بخش دیگر تابع `checkboxbutton init` ها را داریم که مربوط به بخش دوم یعنی ارسال به دیگران است که تعداد آن ها نیز به تعداد مواردی که میخواهیم قیمت آن را داشته باشیم هست. یکی از آن ها:

```
Checkbox(frame۲, text='Dollar',variable=self.dollar).pack()
```

آن را نیز در یک `LabelFrame` برای زیبایی قرار داده شده.

بخش های دیگر `init` را در ادامه توضیح داده خواهد شد.

غیر تابع `init` توابع دیگری نیز داریم که ۱۰ تا از آن ها نقش دریافت اطلاعات سایت و دسته بندی

اطلاعات را دارند از جمله این توابع : `gold_price` , `dollar_price` , `bitcoin_price` , `coin_gold_price` ,

`euro_price` , `bors_price` , `derham_price` , `tether_price` , `iqd_price` , `shasta_price` هستند .

تابع **gold\_price** که نقش آن استخراج قیمت طلا است را بررسی میکنیم:

برای اینکه بتوانیم قیمت های بروز را استخراج کنیم باید از سایت هایی که دائما خودشان را بروز میکنند استفاده کنیم برای مثال سایت [www.tgju.org](http://www.tgju.org) سایتی هست که دائما خوشو آپدیت میکنه.

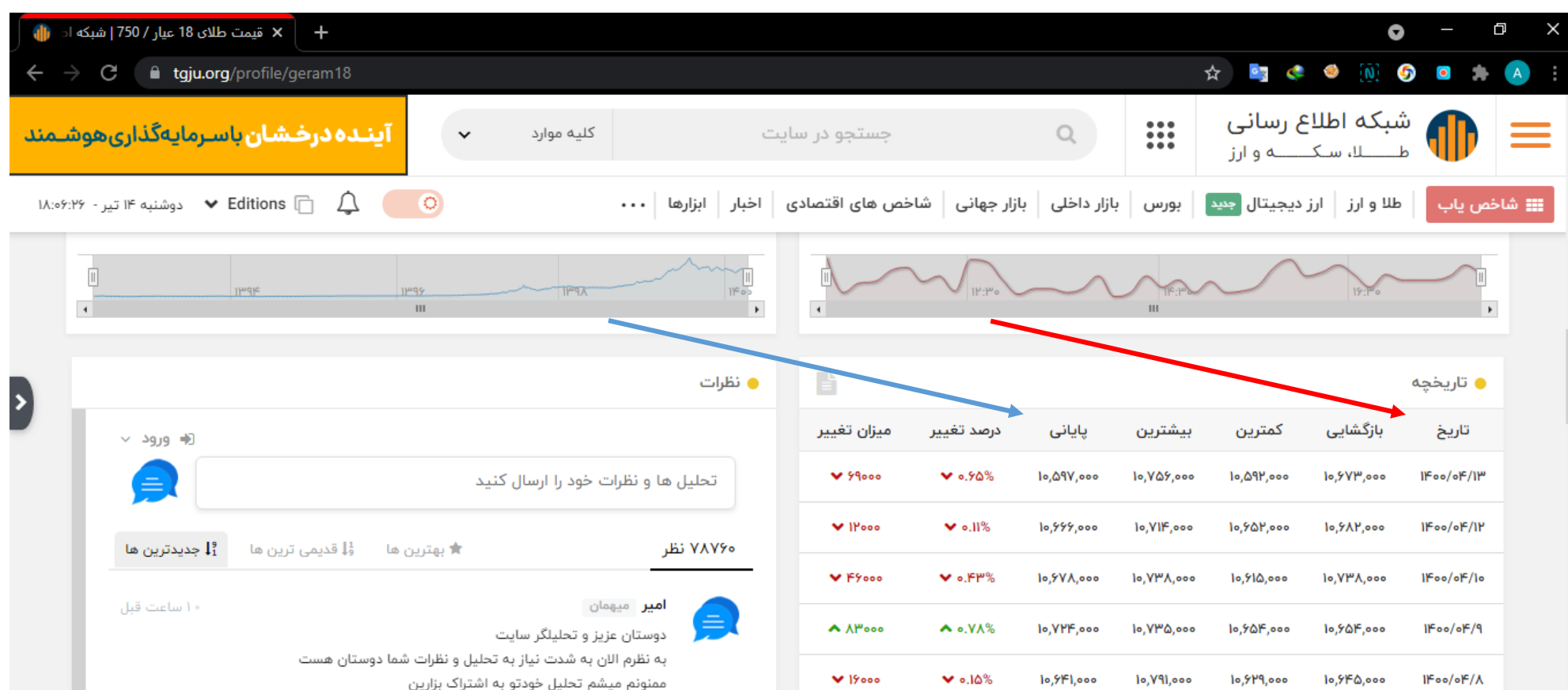
برای اینکار ابتدا میبایست صفحه html مربوط به قیمت طلا را دریافت کنیم با استفاده از متود `get` از کتابخانه `requests` میتوانید اینکار را به سرانجام برسانیم

```
self.gold_price_site = requests.get("https://www.tgju.org/profile/geram18")
```

سپس آن را به تابع `BeautifulSoup` میدهیم

```
soup = BeautifulSoup(self.gold_price_site.text, 'html.parser')
```

سپس با استفاده از متد `select` با استفاده از `css selector` مشخص میکنیم که کجای فایل html را میخواهیم استخراج کنیم که با تصویر نشان داده شده است.



در قسمت تاریخچه ما نیاز به مقدار های تاریخ و قیمت پایانی داریم

برای مشاهده جزئیات با باز کردن تب `inspect` ما میتوانیم جزئیات بیشتری را از صفحه خود داشته باشیم.



با این حلقه لیست dates را به ترتیب با تاریخ ها پر میکنیم.

سپس همین روش را برای استخراج قیمت را انجام میدهیم:

```
for i in range(len(data)):
    child = list(data[i].children)
    prices.append(child[9].text)
```

منتها چون قیمت در تگ td همسطح با td تاریخ و ۵مین ردیف قرار دارد نمیتوانیم از این روش پیش بریم (data.td.td.td.td.td) و باید تمام فرزندان tr را ابتدا استخراج کنیم سپس نهمین تگ را انتخاب کنیم. با این کار ما هردو مقدار های تاریخ و قیمت را در لیست های خود ذخیره کردیم منتها مشکلی که برای لیست قیمت ها وجود دارد این است که قیمت های ذخیره شده در لیست prices به صورت string هستند و نمیتوانیم آن را در نمودار قرار دهیم چون matplotlib به ما value Error میدهد. برای حل این مشکل میبایست قیمت ها را به int تبدیل کنیم و دوباره آن ها را در یک لیست ذخیره کنیم.

```
price_int= []
for i in prices:
    b = i.split(',')
    price_int.append(int("".join( j for j in b)))
```

همانطور که مشاهده میکنید چون در قیمت ها سه رقم سه رقم با ',' جدا شده میبایست ها را split کنیم با split کردن به این روش به ما یک لیست میدهد که در b ذخیره کردیم سپس با متود append آن ها را اول int میکنیم سپس یک string خالی درست میکنیم " بعد با join که یک تابع از نوع string است یک حلقه درست میکنیم که تمام بخش هایی که از کاما جدا کردیم را به این string اضافه کنیم سپس پس از اتمام حلقه یک لیست فقط با اعداد را خواهیم داشت که پس از تبدیل آن به int به مشکل نمیخوریم و آن را راحت به لیست جدید خود اضافه میکنیم به این صورت که لیست جدید ما دارای مقدار int هست.



سپس دو متغیر را برای اینکه بتوانیم از آن ها جدول استخراج کنیم آن ها به یک DataFrame تبدیل میکنیم.

برای اینکار باید ابتدا pandas را import کنیم که ما در ابتدا برنامه این کار را انجام داده ایم.

```
from pandas import DataFrame
```

کامندی که اول برنامه باید اضافه بشه

```
data_frame = DataFrame({'date':dates[:10], 'price':prices[:10] },columns=(['date', 'price']))
```

با این روش یک DataFrame ساختیم

پس از ساخت dataframe در اینجا ما یک قطعه کدی داریم که در صورت وجود یک لیبل که در ادامه توضیح خواهیم داد آن را از بین میبرد(مربوط به جدول اضافه شده در برنامه ما هست چون بعضی از جدول ها بزرگ هستند و بعضی ها کوچک و در صورتی که یک جدول بزرگ نمایش داده شود و پس از آن یک جدول کوچک، یک بی نظمی به وجود خواهد آمد که ما در اینجا قبل از نشان دادن جدول جدید جدول قبلی را پاک میکنیم در صورت وجود)

try:

```
self.label2.destroy()
```

except Exception:

```
pass
```

به بخشی از کد رسیدیم که در آن میبایست نمودار را نمایش بدیم برای این کار همانطور که در ابتدا ذکر کردیم نیاز به کتابخانه matplotlib.pyplot که ما در ابتدای برنامه به این صورت آن را اضافه کردیم که فقط با plt آن را صدا بزنیم.

```
import matplotlib.pyplot as plt
```

برای ساخت نمودار باید به کتابخانه matplotlib.pyplot مقدار را ها اضافه کنیم و سپس نام بردار های آن را مشخص کنیم و سپس مستقیم نمایش بدیم یا ذخیره و بعد آن را نمایش بدیم که من در این برنامه از روش ذخیره و سپس نمایش استفاده کردم. برای باز کزدن تصویر به صورت برای تکینتر از ماژول Image و نمایش آن به صورت لیبل در تکینتر از ImageT استفاده کردیم که از کتابخانه PIL در ابتدای برنامه اضافه شده.



```
plt.plot(dates[5::-1], price_int[5::-1])
plt.xlabel('date')
plt.ylabel('price (Rial)')
plt.savefig('data\\dollar.png', dpi=80 )
image1 = Image.open(r"data\dollar.png")
test = ImageTk.PhotoImage(image1)
label1 = tkinter.Label(self.first, image=test)
label1.image = test
label1.place( x=400, y=40)
```

در ابتدا عکس نمودار ذخیره شده در پوشه data سپس آن را نمایش داده شده همانطور که مشاهده میکنید آن را به صورت یک لیبل در آورده ایم که در این صورت میتوانیم مکان آن را مشخص کنیم که در شکل مشاهده میکنید که self.first را انتخاب کرده ایم به این معنا که در tab اول نمایش داده شود.

در بخش بعدی تبدیل DataFrame به جدول است برای بتوانیم این کار را انجام بدیم باید از ماژول dataframe\_image استفاده کنیم که بنده آن را به صورت مخفف dfi در ابتدای برنامه اضافه کردم.

برای این کار ابتدا میبایست dataframe خود را به یک تصویر تبدیل کنیم و سپس با ماژول Image آن را باز میکنیم و به نمایش میگذاریم

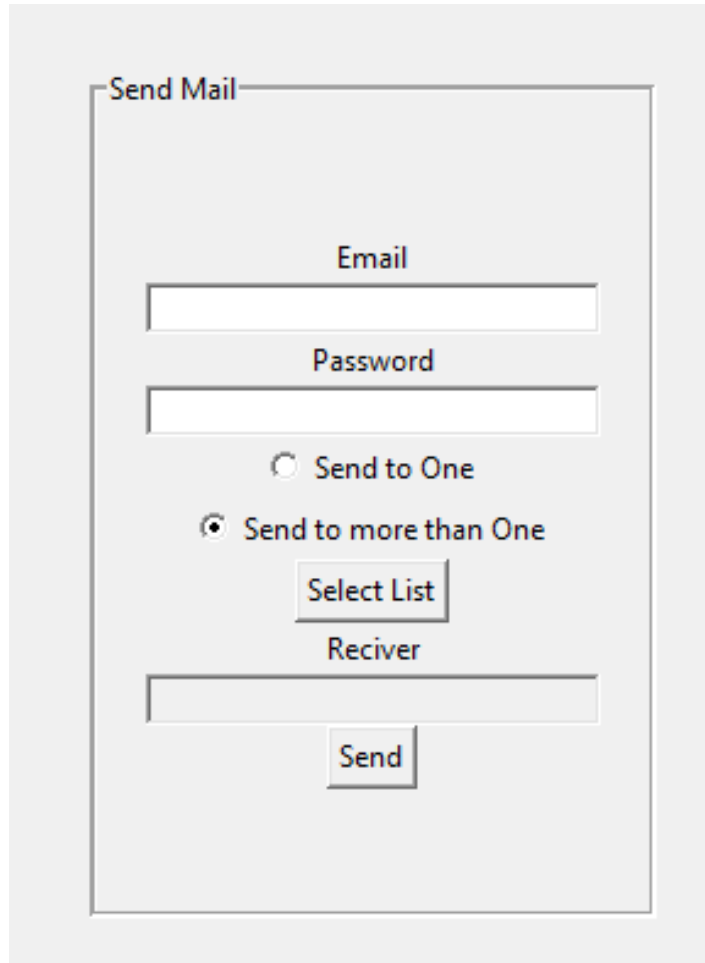
```
dfi.export(data_frame,'data\\gold_dataframe.png')
image2 = Image.open('data\\gold_dataframe.png')
test2 = ImageTk.PhotoImage(image2)
self.label2 = tkinter.Label(self.first,image=test2)
self.label2.image=test2
self.label2.place(x = 200,y =40)
```

و در آخر باید plt.close() را اضافه کنیم تا plt خالی شود تا در توابع بعدی به مشکل نخوریم.

و در آخر نیز قیمت و تاریخ را بازگردانده ایم این بخش مربوط به دو تابع ارسال sms و mail هست تا مقداری که بازگردانده شده را ارسال کند.

```
return(f"Gold price at : {dates[0]} is : {prices[0]} (Rial) ")
```

## تابع `send_mail` در بخش دوم برنامه (`send to others`)



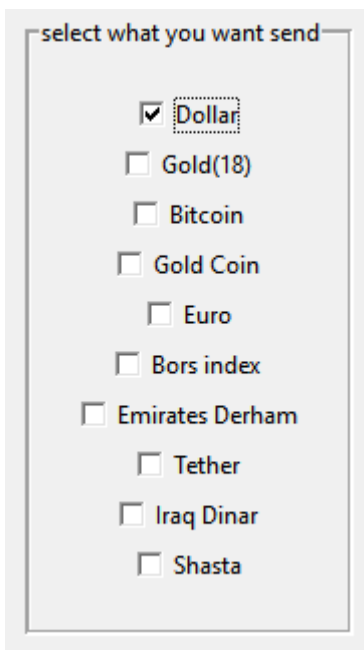
برای ارسال اطلاعات از طریق ایمیل ابتدا باید مطمئن شویم که کاربر میخواهد به یک نفر ایمیل ارسال کند یا بیش از یک. برای این کار باید محتوای متغیر `one_or_more` را دریافت کنیم محتوای آن یک بود باید ایمیل گیرنده را از بخش `Entry` به نام `reciver` دریافت کنیم در غیر اینصورت باید ایمیل هارا از فایل `txt` که دارای بیش از یک ایمیل هست را دریافت کنیم.

در بخش بعدی باید مطمئن شویم که ایمیل به درستی درج شده باشد به همین دلیل از این شرط استفاده کردیم:

```
if '@' in self.entry_for_reciver.get() and '@' in self.email_sender.get() and self.password.get() != '':
```

یعنی در صورت وجود '@' در دو بخش `Entry` های ایمیل گیرنده و ارسال کننده و عدم خالی بودن بخش `password` ایمیل را ارسال کنید .

پس از آن باید بدانیم که کاربر قیمت کدام بخش ها را میخواهد ارسال کند((checkbox))



برای اینکار برای هر کدا از آن ها یک شرط گذاشتیم که در صورت اینکه مقدار آن ها یک بود به معنای یک بودن آن است و آن را به یک لیست اضافه کردیم و در نهایت اطلاعات را با استفاده از کد زیر به تابع ارسال ایمیل ارسال میکنیم

```
self.mailSender(self.email_sender.get(),self.entry_for_reciver.get(),self.password.get(),prices)
```

که در تابع mailSender ما ۴ مقدار را دریافت میکنیم یکی mail\_sender هست که ارسال کننده ایمیل و دیگری mail\_reciver و password\_sender و prices که لیستی از قیمت هایی است که کاربر آن را تیک زده و خواهان ارسال آن است.

برای ارسال ایمیل ابتدا باید مطمئن شویم که اطلاعات به درستی در entry باکس ها قرار گرفته شده اند.

تابع `send_with_phone` : تابعی برای ارسال قیمت ها با استفاده از sms است.

در این تابع ابتدا باید چک باکس های علامت خورده را در یک لیست ذخیره کنیم تا متوجه شوید کاربر کدام قیمت ها را میخواهد ارسال کند.

```
prices = []
if self.gold.get():
    prices.append(self.gold_price())
if self.dollar.get():
    prices.append(self.dollar_price())
if self.bitcoin.get():
    prices.append(self.bitcoin_price())
if self.coin.get():
    prices.append(self.coin_gold_price())
if self.bors.get():
    prices.append(self.bors_price())
if self.euro.get():
    prices.append(self.euro_price())
if self.derham.get():
    prices.append(self.derham_price())
if self.tether.get():
    prices.append(self.tether_price())
if self.iraq.get():
    prices.append(self.iqd_price())
if self.shasta.get():
    prices.append(self.shasta_price())
```

مقدار هر کدام که یک بود به این معناست که چک باکس تایید شده است و توسط کار بر تیک خورده است. پس به لیست ما اضافه شده.

سپس لیست شماره ها را از فایل txt دریافت میکنیم و تک تک آن ها را به کاربر ارسال میکنیم.

در اینجا از ویژگی multithreading استفاده کردیم تا کار را سریع تر پیش ببریم.

```
with open(self.filename,'r') as f:
```

```
    first = f.readline()
```

```
    while True:
```

```
        if first :
```

```
        threading.Thread(target=self.send_message_with_ghasedak,args=(first.split('\n')[۰],prices,)).start()
```

```
        first= f.readline()
```

```
    else: break
```

که خود این خاصیت باعث میشود به صورت موازی مقدار ها را به تابع

self.send\_message\_with\_ghasedak ارسال کند که باعث صرفه جویی و انتظار کمتر کاربر میشود.

برنامه شامل بخش های دیگری نیز هست برای مشاهده به مسیر پروژه به آدرس زیر مراجعه نمایید:

<https://github.com/alianjo/price-to-send>