

CIT244 Python II Program 4

Program 4: more on wx.python, web services, & databases

We start with an sqlite3 database named `tech_stocks.db`. The database has a table named `dow_stocks`, which contains data on 9 of the 30 stocks that make up the Dow Jones Industrial Average. The table contains a primary key named `sid`, and columns for the company name, the exchange the stock trades on, the stock symbol, the number of shares purchased, and the price paid when the individual shares of that stock were purchased. Do not change these column names: `sid`, `company`, `exchange`, `symbol`, `shares`, `purchase_price`. Note the stock symbol for Apple is AAPL. In fact, make no modifications to this database.

An image is shown below.

The screenshot shows a SQLite database viewer interface. On the left, a tree view lists several databases: college (SQLite 3), autobazaar (SQLite 3), db3 (SQLite 3), speeding_tickets (SQLite 3), counties (SQLite 3), trades (SQLite 3), ticketsPY (SQLite 3), and tech_stocks (SQLite 3). The 'tech_stocks' database is selected, and its 'Tables' folder is expanded, showing the 'dow_stocks' table. The main area displays the 'dow_stocks' table in 'Grid view'. The table has 9 rows and 6 columns: sid, company, exchange, symbol, shares, and purchase_price. The data is as follows:

	sid	company	exchange	symbol	shares	purchase_price
1	1	3M	NYSE	MMM	100	157.5
2	4	Apple Inc.	NASDAQ	AAPL	100	102.4
3	5	Boeing	NYSE	BA	200	157.2
4	6	Caterpillar Inc.	NYSE	CAT	200	162.35
5	8	Cisco Systems	NASDAQ	CSCO	100	35.75
6	13	Honeywell	NYSE	HON	100	180.25
7	14	IBM	NYSE	IBM	300	111.45
8	15	Intel	NASDAQ	INTC	200	66.1
9	20	Microsoft	NASDAQ	MSFT	100	177.8

sqlite3 database name: `tech_stocks.db`

table name: `dow_stocks`

Download a zipped copy of the database from the course website.

[tech_stocks.zip](#)

In a real application you would have code that allows updating or inserting new values as stocks are bought and sold. A column holding dates would also probably be handy. There might likely be other tables as well. But here we are not trying to make the perfect app. We just want to develop code to get current info and compare it to saved data.

What we would like to do is make a daily query of a web service that offers free stock quotes, and have our program compare the current price for each stock with our original purchase price. Then, using the number of shares purchased, we calculate and display the current profit or loss for each stock. We also want to know the net profit or loss after looking the results of all 9 of the individual

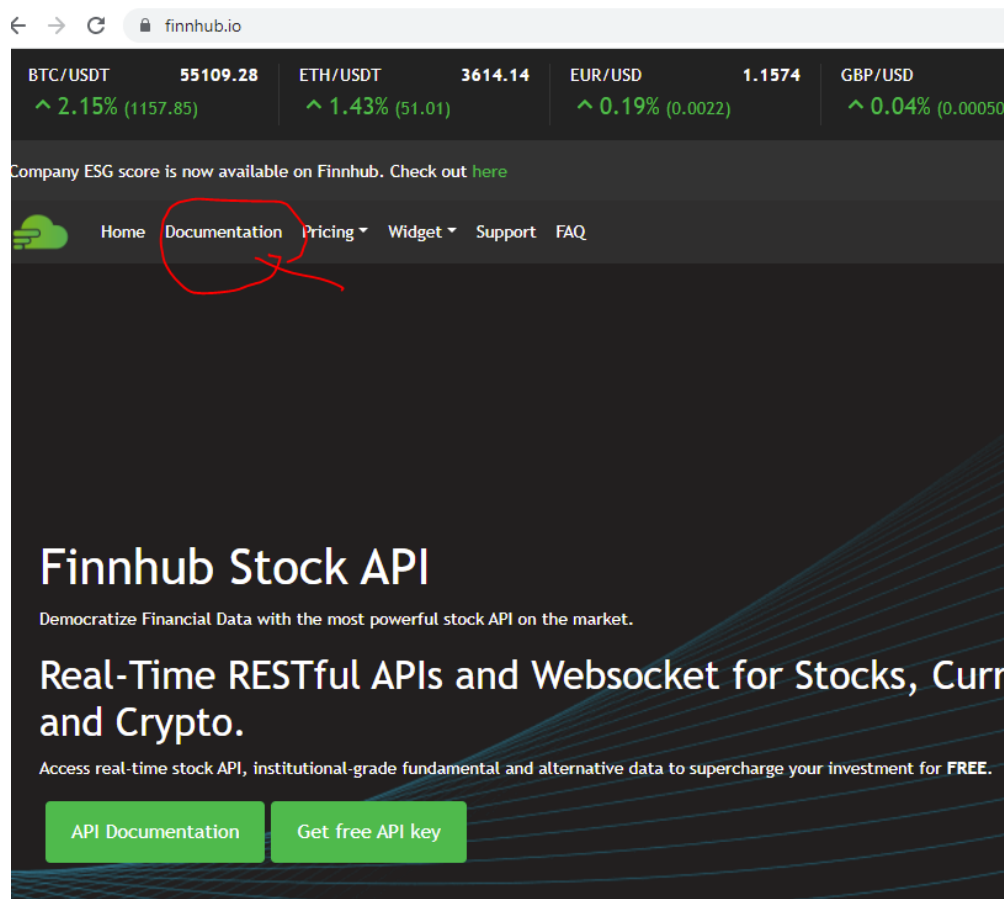
stock values to see our overall gains or losses. There are a very large number of web services dealing with financial and stock market issues. We will use a free service named `finnhub.io`.

The Details

We will make a *desktop app* using wxPython that queries a web service named *finnhub.io* for data. We will combine this data with some of the data in our `tech_stocks` database. In order to make Ajax/JSON requests to `finnhub.io` and receive data, we first need to do a couple of things.

1. go to `finnhub.io`, click the button that says "Get free API key".
2. get your API key (or, sometimes it is called an access token). Store it somewhere easy to find.

This key will be attached to any requests you make for data. Free access to Finnhub data is allowed as long as you don't make any more than 30 API requests per second. The key is mainly used by finnhub to ensure you do not exceed this limit. We will have no trouble making less than 30 request per second. Here is the default page for finnhub. Note the green buttons at the bottom of the image shown. Get your access key.



If you click the API Documentation button, you'll find a ton of info on the services offered by finnhub. But we are mainly interested in getting an up-to-date quote for each of our 9 stocks. The section is named **Quote**. The information on how to do that is given a little more than half way down the documentation page, and in fall of 2020 this info looks like this next image. The page may have changed some, they may not show the example code as given below, but they will show an example response. The response, after being deserialized from JSON, is a python dictionary of stock values. Keys of c, h, l, o, pc, and t, as shown in the image below. Here these keys have values of closing price (c), high for the day (h), low for the day (l, lower case L), the opening value of the stock (o), and the previous close (pc). They use Apple in this example.

Quote

Get real-time quote data for US stocks. Constant polling is not recommended. Use websocket if you need real-time update.

Real-time stock prices for international markets are supported for Enterprise clients via our partner's feed. [Contact Us](#) to learn more.

Method: GET

Examples:

`/quote?symbol=AAPL`

`/quote?symbol=MSFT`

Arguments:

symbol **REQUIRED**
Symbol

key: value

"c": 123.4 closing value

"h": 123.4 high

"l": 123.4 low

"o": 123.4 opening value

"pc": 123.4 previous close

Sample code

Python

```
1 import requests
2 r = requests.get('https://finnhub.io/api/v1/quote?symbol=AAPL&token=bu7
3 print(r.json())
4
```

Sample response

```
1 {
2   "c": 261.74,
3   "h": 263.31,
4   "l": 260.68,
5   "o": 261.07,
6   "pc": 259.45,
7   "t": 1582641000
8 }
```

```
r.requests.get('https://finnhub.io/api/v1/quote?symbol=AAPL&token=abc7youraccesscode4h5')
```

Response Attributes:

In the JSON response you may also have a d key (change), or a dp key (percent change). Sometimes these keys are there, sometimes not.

Test

As a preliminary test you should copy the following URL, replace the last part with your actual access token, and paste it into your browser address bar. Hit enter. You should get a JSON response back for Caterpillar in JSON format.

```
https://finnhub.io/api/v1/quote?symbol=CAT&token=youraccesstoken
```

In our Python code we will need to use a URL like this with the requests library. The structure of a request looks like this next code chunk when using Python. This is an example python request which should print out the returned

information in a form that has been deserialized; it should be a regular python dictionary, accessible using the methods we normally use to manipulate dictionary data. The access token used here is a fake and will not work, so either remove it or use your own key if you want to test it out in the python shell or whatever IDE you use.

```
res =
requests.get('https://finnhub.io/api/v1/quote?symbol=AAPL&token=youraccesstoken')
res = res.json()      # convert response from JSON to Python
```

This part of the URL is the base.

`https://finnhub.io/api/v1/quote?`

And to that we need to attach, using the Apple stock symbol as an example, the following

```
symbol=AAPL
```

and along with an ampersand, you would add the name=value pair for your access token. The token below is fictional and will not work. Your api key (or token) is usually a mix of 20 random-looking characters.

```
&token=12g45amc0123      # this is not a real access key, do not use
```

As we said, initially the response comes back in JSON format. As we know, the following **.json()** python method deserializes the data in the variable named *reqData* from the JSON format to a regular python dictionary.

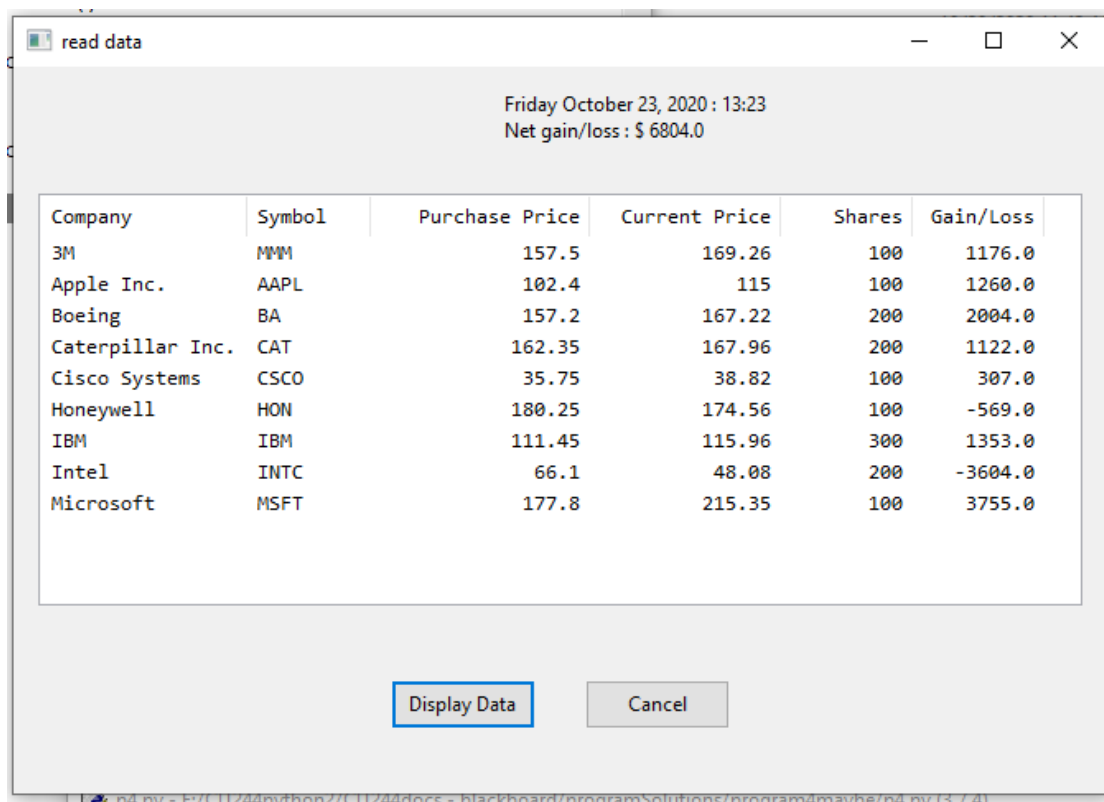
```
dictionary_data = reqData.json()
```

Of the response values, we only need the current value of the stock. But shown below is a minimum number of lines of code that would make a request (assumes you have pip installed *requests* at some point in the past). In this case we want the IBM stock values. This was done in October 2020. The API key has been blurred. As we said, the data comes back to use as JSON by default.

We concatenate the stock symbol into the URL, a technique you should find useful in this assignment.

This next image is what it looked like when I clicked the Display Data button on Oct 23, 2020. As an example calculation, we can see that we originally bought 100 shares of 3M for \$157.50 per share, which is stored in our database. We queried finnhub on Oct 23 and got a current price 169.26. The overall gain or loss for this stock would be 100 shares times (current price - purchase price).

This calculation will be negative (a loss) if current price is less than purchase price. $\text{gain} = 100(169.26 - 157.5) = 1176$.



Friday October 23, 2020 : 13:23
Net gain/loss : \$ 6804.0

Company	Symbol	Purchase Price	Current Price	Shares	Gain/Loss
3M	MMM	157.5	169.26	100	1176.0
Apple Inc.	AAPL	102.4	115	100	1260.0
Boeing	BA	157.2	167.22	200	2004.0
Caterpillar Inc.	CAT	162.35	167.96	200	1122.0
Cisco Systems	CSCO	35.75	38.82	100	307.0
Honeywell	HON	180.25	174.56	100	-569.0
IBM	IBM	111.45	115.96	300	1353.0
Intel	INTC	66.1	48.08	200	-3604.0
Microsoft	MSFT	177.8	215.35	100	3755.0

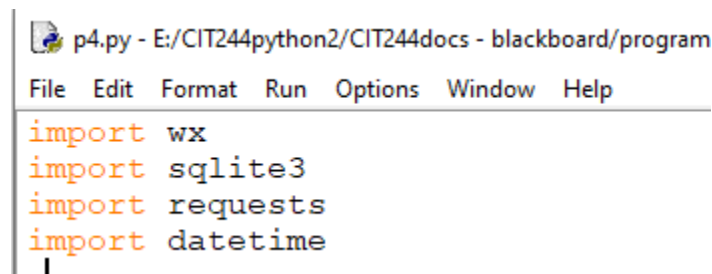
Display Data Cancel

Generally, you need to query the database table to get the original data. Then you're gonna need a loop to make the 9 finnhub queries that get the returned current prices, then make the proper calculations and add up the 9 gain/loss values to get the Net gain/loss. This loop is where most of the calculations would likely be performed. Be careful not to make an infinite loop. You only get 30 requests per second. After looping, then display the results in the GUI for your user.

Prerequisite: If you have not already installed the requests library in an earlier lesson, then before running any code open the command prompt and install requests. If you did the examples in from the lecture notes you may already have installed requests.

```
pip install requests
```

You're gonna need to import `wx`, `sqlite3`, `requests`, and we need to be able to get the date, which is not included in the finnhub data set, so you will also need to import `datetime`



```
p4.py - E:/CIT244python2/CIT244docs - blackboard/program
File Edit Format Run Options Window Help
import wx
import sqlite3
import requests
import datetime
|
```

We've not done much with Python dates, but the Internet is full of examples. Here's a code hint. We need to import *datetime*, as shown above, and if so the first line below gets the information for NOW, that is, the current date and time. The second line formats the date as day-of-the-week, month, day, year, hours (on the 24 hr clock), and minutes.

```
x = datetime.datetime.now() # date and time
date = x.strftime("%A %B %d, %Y : %H:%M")
```

For full credit:

- Use either sizers or absolute positioning; you're choice. your layout does not need to look exactly like mine, but it does need the same widgets and they need to be visible when the window opens. the list control needs to be large enough to display the columns without a horizontal scroll bar showing up.
- the values are dollars, so round any calculations to no more than 2 decimal places so there are no numbers that look like 345.3333222211123344
- you need a loop to send the 9 stock requests. in order to have a loop send your requests to finnhub, you will need an API key. if you don't want me to see your api key, you can type wxyz in for the access token before sending me your code, but leave the rest of the code intact so I only have to substitute my key in place of yours. and don't lose your access key.
- to make this work your code will have to query the database and retrieve the company, symbol, purchase price, and number of shares in order to display this data along with the finnhub values. Do *not* hard code the dow_stocks information in your program.
- your code will need to work with my copy of the database.

Start early. As usual, there is an example in the lecture notes that should give you a good start. Let me know if there are questions.

BIG Hint: You need a loop, a loop that reads a line of data in from the database, then makes a request to the web service for the current price of that stock, uses that stock price to calculate profit and loss, then puts everything in the list control for that row. There are other things to do as well. Here's kind of an outline.

```
#connecte to stocks db, fetch the rows of data

#loop thru each row of the stocks table.
    # get the stock symbol for the stock in this row
    # concatenate the request to the web service for that stock
    # get the current price for the stock
    # calculate profit or loss
    # append info to the list control
```

For a not-perfect generalization, assuming the access token you requested is stored in a variable named `tok`. `r[1]` would be the stock symbol

```
for r in rows:
    url = 'https://finnhub.io/api/v1/quote?symbol='+ r[1] + '&token=' + tok
    response = requests.get(url)
    apiData = response.json()
    stockGainLoss = calculateGainLoss
    overallGainLoss += stockGainLoss
    self.list.Append(data-for-list-control)
```

Grading will follow this approach.

- 100% program runs perfectly with all required items
- 90% program meets all requirement but doesn't always produce the correct output
- 80% program runs without errors, but is missing some requirement.
- 70% program had a good start, most but not all of the code was correct, had one or more fatal errors.
- 40-60% or below. Not very close, many parts missing, would not run, probably a late start.
- 0% no submission or code was not the student's work.

How To Submit Your Program

I have the database, you don't need to send yours. I just need the code. You also know how to submit your program by now.

If you have questions let me know: mark.prather@kctcs.edu.