《Introduction to programming》

The Second Semester of the 2018-2019 Academic Year

June 9th, 2019

Group Project Report

Stocksale

STUDENT NAME	STUDENT ID
Eugster Robin Dennis	7161209900914
Ikebe Yuta	716120290036
Sugaya Yusuke	716120290040

Abstract

Keywords: Stock-market, API, Automatic Messaging

Table of Contents

1. Introduction

Stocksale is a program that seeks to apply the team's (Hackermen) newly acquired programming-skills in the field of finance.
More specifically, it aims to be a helpful tool for students that want to get started in trading.
Among business students, many develop an interest in the stockmarket sooner or later.
This paper identified two major problems that are characteristic but not limited to students and aims to be a solver of this problems.
Firstly, students usually possess a limited budget which is the key hindrance to buy shares.
Most of the broker's fees will surpass the benefits that can be reaped on the stock-markets with low volume purchases.
The second problem Hackermen identified is the scarcity of time. Being a professional in the markets and being able to build a lasting fortune with company shares usually requires a fundamental analysis of numerous corporations as well as the sector as a whole.
Students lack this time due to their tight course schedule and a flooding of assignments.

Most brokers already provide a similar service, that allows their customers to set caps and floors on there positions and interests, but with one major difference: As soon as the price target is hit by the market, the buying (or selling) order will be automatically executed, without asking the investor a second time if the set target still is the desired one.
This might work for affluent long-term investors but can cause problems for students, since their initial buying plans can be disrupted by an unexpected expenditure, such as a malfunctioning phone or laptop.
Hackermen want to provide an alternative for this segment and are not aware of a comparable service.

As already stated, Stocksale will be a tool that monitors the market's in real-time and informs the user via an automated message, if the price of his dream-company's shares will reach his budget.
The following sections will evaluate the development process, the program's functions and the limitations in detail.

2. Development process

The development process of our programe can be broken down into … steps:

Step 1: Selection of a financial instrument

Step 2: Selection of the price, which the user is willing to pay

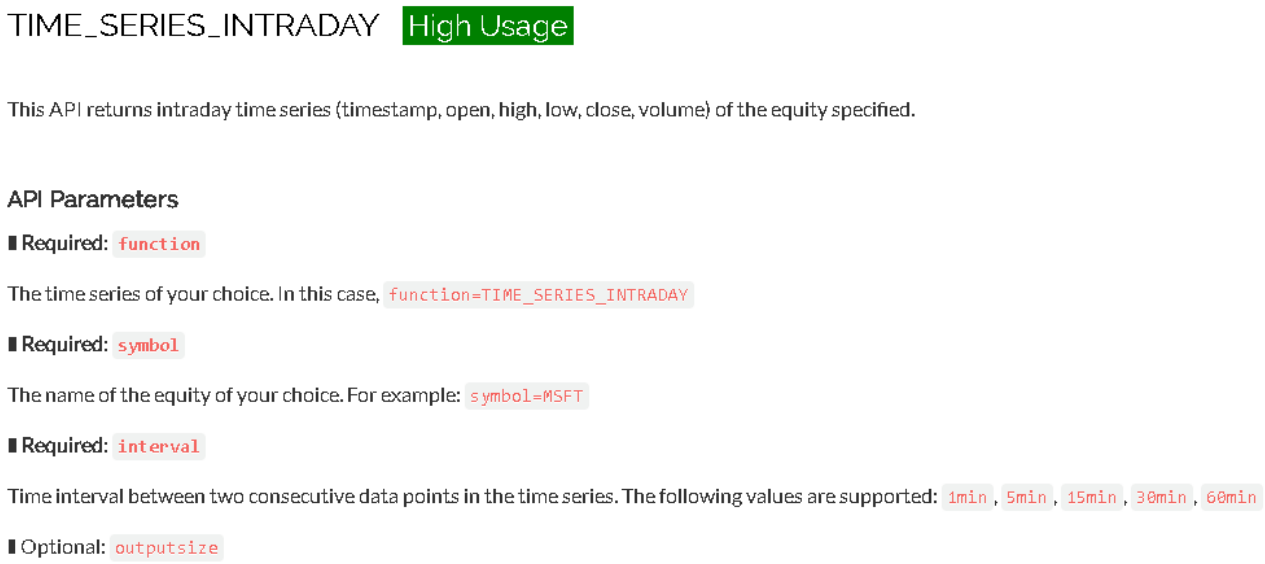
Step 3: Scanning the markets

Step 4: Information once the target price matches the actual price

Step 5: Users decision of buying or passing

Each of this step will be elucidated and the corresponding lines of codes will be exemplified. (The full code can be accessed on github(gist): https://gist.github.com/LordUdonius/3e3ce6af46f7bcbdcde23206a438f739 , which functioned as Hackermen's primary working tool).

Before any company or budget can be selected, an efficient way of accessing financial data must be found. An easy and userfriendly way to receive such data is the use of an API (Application Programming Interface) and Alpha Vantage (https://www.alphavantage.co/) was the most promising amongst various competitors.
After applying for an API access key, Alpha Vantage provides actual and historical stock, FOREX and cryptocurrency market data. They cover the most important global exchanges and provide a detailed documentation about the available functions (*see figure 1).*



(*Figure 1*)

After importing and embedding the functions, the reliability could be tested. Among the possible features, we limited our research on the usage of the "TIMESERIES_INTRADAY" API which best fulfills our purpose (*see figure 2*).

```
1   from alpha_vantage.timeseries import TimeSeries
```
(*Figure 2*)

This API allows us to screen the markets every minute to derive the most recent prices of whatever share we desire to buy (*see figure 4*).

```
def getStockPrice():
  '''This function screens the market for the current price of the priorly selected stock'''
    current_time = time.time()
    ts = TimeSeries(key='YOUR_API_KEY', output_format='pandas') #define API
    data, meta_data = ts.get_intraday(symbol=company_selector(company),interval='1min', outputsize='full') #get API data for 1min
    data2 = data.tail(1)    #get most recent Stock price
    closePandas = data2["4. close"]
    closePrice = closePandas[0]
    #pprint(closePandas[0])     #test print
    #print(type(closePrice))    #test print
    return closePrice
```

(*Figure 3*)

After successfully deriving the first set of data, this task was to convert the data type from json (*see figure 4*) to pandas, which is a necessary requirement to have the data set ready for further calculations.

```
"Time Series (5min)": {
    "2019-06-07 16:00:00": {
        "1. open": "131.1900",
        "2. high": "131.5000",
        "3. low": "131.1800",
        "4. close": "131.4500",
        "5. volume": "1302062"
```

(*Figure 4*)

Eventually, we could create our own function, that allows the user to select his financial instrument of interest. This company's market information could be successfully displayed in the panda data type subsequently.

```
company = str(input("which company's shares do you want to buy? "))
def company_selector(company):
  '''this function works as a symbol translator to later feed the API with the necessary string to receive the real-time data.'''
    symbol = 0
    while symbol == 0:
        if company == "ABB":
            symbol = "ABBN"
        elif company == "Apple":
            symbol = "AAPL"
        elif company == "Alcon":
            symbol = "ALC"
        elif company == "Credit Suisse":
            symbol = "CSGN"
        elif company == "Geberit":
            symbol = "GEBN"
        elif company == "Givadaun":
```

(Figure 5)

For the sake of our project, we limited the available financial instruments on the shares that are being traded in the Swiss Market Index (*see figure 5*), since this is a comprising list of the 20 big corporations in Switzerland and this sample is sufficient to test the successful execution of our program. Adding further companies or different financial instrument is only manual work and does not require any additional programming knowledge.
What must been made sure of is that the user can enter the listed name of the company, while our function transforms the name to the listed symbol, since this is the string that allows the API to search for the necessary data.

Since the average business student is not aware of the symbol, under which a public company is traded on the market, our program executes a transformation from the name to the symbol and saves valuable time.

The next task was to let the user enter his desired price, at which a single share will be affordable for him/her. Since brokers, such as Robinhood trading (https://www.robinhood.com/) enable trading without any transaction fees, buying single shares is possible in today's investing environment.

```
targetPrice = float(input('what is your Target Prices?'))
emailaddress=str(input("Please enter your email address"))
```

(Figure 6)

After deriving the target price and the E-Mail address of the user (which will be later used as a means to receive the message of the market data) (*see figure 6*), the "compareToTarget()" function will compare the inserted price with market (*see figure 7*).

```
def compareToTarget():
  '''This function compares the market price with our targeted price. If the market price is below our targeted price, it will send
    closePrice = getStockPrice()
    print(closePrice)
    if(closePrice > targetPrice):
        print("Close Price is over your target price!")
```

(Figure 7)

Since the prices must not just once be compared but in a regular time interval (which is being set on one minute), we had to include the following lines of codes:

```
def func2():
  '''this function will keep the above functions running to do a price comparison every minute.'''
    current_time = time.time()  #If you want to add function, you can use this func. watch url before i wrote.
    print('func2')
"""

funcs = [timekeep, compareToTarget]    #Sets of all functions you want to run same time.
threads = []
current_time = time.time()

for i in range(1, 3):
    for func in funcs:
        t = threading.Thread(target=func)
        t.start()
        threads.append(t)

    for t in threads:
        t.join()
    print(str(i) + 'minutes after: ' + str(time.time() - current_time))     #display how much time you spend before run this progra
```

(Figure 8)

Eventually a function had to be written, which specified the message text, once the price of the market and the customers target price are congruent.

```
def compareToTarget():
  '''This function compares the market price with our targeted price. If the market price is below our targeted price, it will send
    closePrice = getStockPrice()
    print(closePrice)
    if(closePrice > targetPrice):
        print("Close Price is over your target price!")

        gmail_user = 'hackermen6619@gmail.com'
        gmail_password = 'hackermen.66.19'

        sent_from = gmail_user
        to = [emailaddress]
        subject = 'the stock reached your target price'
        body = """Dear user,

the stock reached your target price.
Please check it out on our program!
this is the link to your broker: https://robinhood.com/signup/?utm_source=google&utm_campaign=1372881561&utm_content=542216466658ut
If you want to stop mail delivering service, send an email to hackermen6619@gmail.com and we will stop to inform you immediately.


Hackermen"""

        email_text = """\
        From: %s
        To: %s
        Subject: %s

        %s
        """ % (sent_from, ", ".join(to), subject, body)

        try:
            server = smtplib.SMTP_SSL('smtp.gmail.com', 465)
            server.ehlo()
            server.login(gmail_user, gmail_password)
            server.sendmail(sent_from, to, email_text)
            server.close()

            print('Email sent!')
        except:
            print ('Something went wrong...')
    else:
        print("close Price isn't over your target price.")

"""
def func2():
  '''this function will keep the above functions running to do a price comparison every minute.'''
    current_time = time.time()  #If you want to add function, you can use this func. watch url before i wrote.
    print('func2')
"""
```

(Figure 9)

The whole procedure of the code will be exemplified in the additional video.

3. Limitations & Improvement areas

There are various limitations to our program at the current stage.
One is the limited amount of companies that can be selected, which as already mentioned can be expanded quickly.

Another one is the fact that the user still needs to do all the search of information on external platforms. With the inclusion of additional Alpha Vantage API's, the amount of relevant data

can be increased and also included in the mail. Not only the price is important but of even higher relevance are current company news and events.

Lastly, once an e-mail has been sent because the targeted price has been reached, the user will get e-mails every minute without being able to stop the flow of e-mails on his own. The possibilities of sending us a “stop” message or categorizing our mails as “spam” are two options for him to stop the flow of mails, but while the prior requires effort from as, the ladder is an even more undesirable outcome.

This features can be included at a later time and the program can be embedded in a homepage to make the program accessible for students and to generate true added value.

4. Conclusion

Stocksale’s initial aim of becoming a program that scans the market and informs the user as soon as his budget is reached, could be reached.
This was only possible due to the teachings of our instructor, mister Bao Yang, who learned us to use Python and showed us what is possible with this language, and the briefings of Nicholas Breitkreuz and Kenta Sugaya who showed us how to apply API’s respectively automated messages.

5. References

The sources that are being used in our report stem entirely from our source code (https://gist.github.com/LordUdonius/3e3ce6af46f7bcbdcde23206a438f739), our acquired knowledge in the course *Introduction to programming* and the upcoming presentation.

6. Disclaimer

Since there might occurs a problem in the execution of the code in the above Gist, the following link: https://gist.github.com/hackermen6619/f76ef0df286f62440c3b330127a0c7fb leads to the program that worked and this code was being used in the video. It is important to turn the VPN due to Gmail being a Google service.