Documentation of Project

JP Morgan Chase & Co.

Software Engineering Virtual Program

Name: Mahesh Ratnaparkhe
Contact Details: mahiratnaparkhe4052@**gmail.com**

InsideSherpa

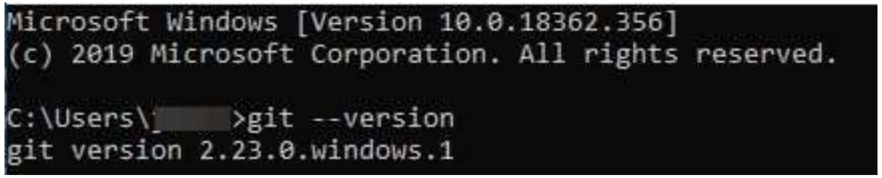
JPMorgan Chase Software Engineering Virtual Experience

Setting up your Windows for the JPMorgan Chase program

Module 1 - Interface with a stock price data feed

Local Setup (Windows)

- First you must have git installed in your system. Git is used by most programmers today to collaborate with code/software projects.To install git, follow this quick guide. You know you have installed successfully when you get this output on your command line (cmd). (*any git version should suffice but the latest is recommended)*



```
Microsoft Windows [Version 10.0.18362.356]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\[redacted]>git --version
git version 2.23.0.windows.1
```

Local Setup (Windows)

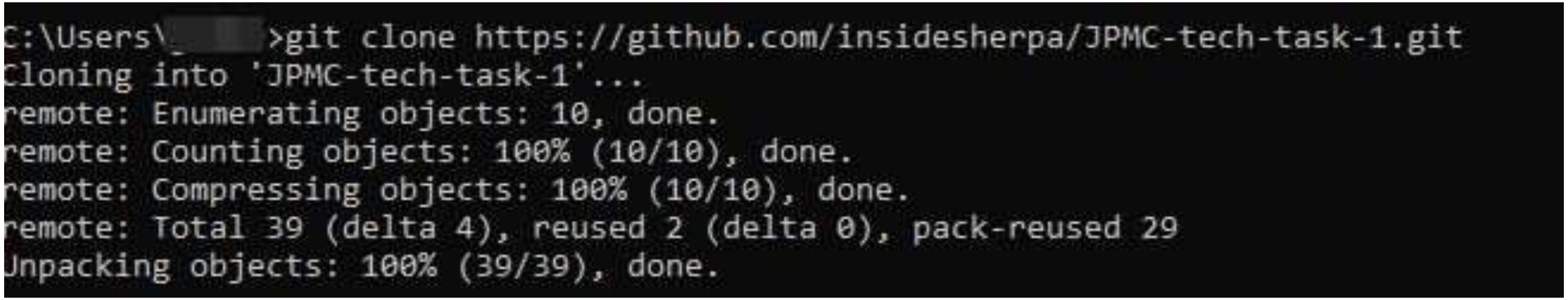
- Once you have git installed, you need a copy of the application code you'll be working with on your machine. To do this, you must execute the following command on your terminal:

```
git clone https://github.com/insidesherpa/JPMC-tech-task-1.git
git clone https://github.com/insidesherpa/JPMC-tech-task-1-py3.git
```

- This command will download the code repositories from github to your machine in the current working directory of the terminal you executed the command in. Downloading the 2 repositories above will give you options later

Local Setup (Windows)

- You'll know you cloned successfully if you have the copy of the application code on your machine:



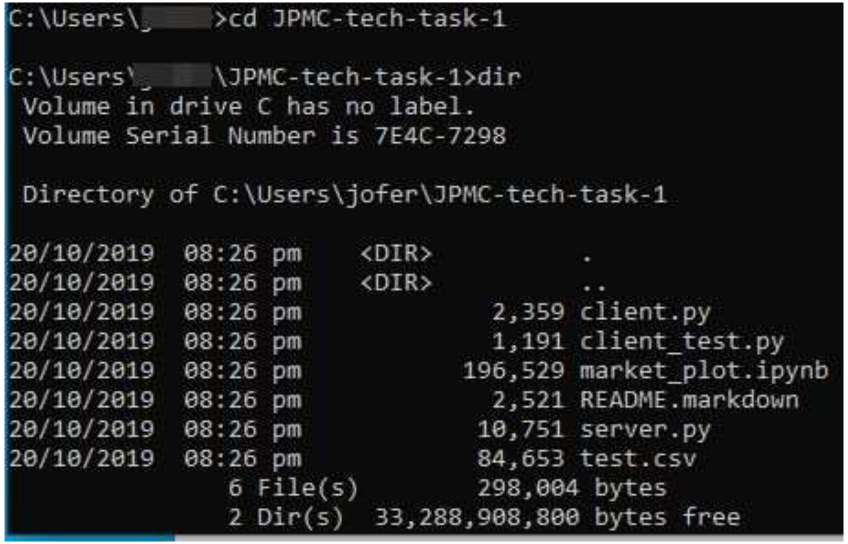
```
C:\Users\[redacted]>git clone https://github.com/insidesherpa/JPMC-tech-task-1.git
Cloning into 'JPMC-tech-task-1'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 39 (delta 4), reused 2 (delta 0), pack-reused 29
Unpacking objects: 100% (39/39), done.
```

note: the image above just does not contain the other repository but it should if you did the previous slides and execute the ***dir*** *command. `dir` will list the contents of the current directory*

Local Setup (Windows)

- To access the files inside from the terminal, just change directory by typing:

`cd JPMC-tech-task-1`



```
C:\Users\[redacted]>cd JPMC-tech-task-1

C:\Users\[redacted]\JPMC-tech-task-1>dir
 Volume in drive C has no label.
 Volume Serial Number is 7E4C-7298

 Directory of C:\Users\jofer\JPMC-tech-task-1

20/10/2019  08:26 pm    <DIR>          .
20/10/2019  08:26 pm    <DIR>          ..
20/10/2019  08:26 pm             2,359 client.py
20/10/2019  08:26 pm             1,191 client_test.py
20/10/2019  08:26 pm           196,529 market_plot.ipynb
20/10/2019  08:26 pm             2,521 README.markdown
20/10/2019  08:26 pm            10,751 server.py
20/10/2019  08:26 pm            84,653 test.csv
               6 File(s)        298,004 bytes
               2 Dir(s)  33,288,908,800 bytes free
```

note: `cd` means change directory (more on cd)

*note: The image on this slide just does not contain the other repository but it should if you did the previous slides and execute the **dir** command.*

note: If you choose to work using python3 and your system has version python3 or above instead of python2.7.x, then choose to go into the other repository you downloaded instead. (otherwise, use the other repo above)

`cd JPMC-tech-task-1-py3`

Local Setup (Windows)

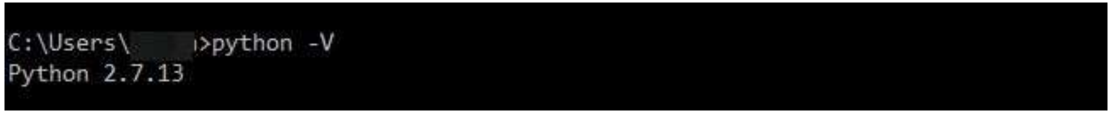
- To clarify, you're only supposed to work on one of the repositories you cloned / downloaded into your system. It all depends on what Python version you primarily use.

- Python is just a scripting / programming language we developers use quite often in the field. This application you'll be working on uses it.

- We'll discuss checking / installing Python in your system in the following slides

Local Setup (Windows)

- Next, you'll need to have Python 2.7 or Python3 installed on your machine. Follow the instructions here (for python2), or here (for python3) You can verify this on your command line (cmd) if you get a result like:



```
C:\Users\[redacted]>python -V
Python 2.7.13
```

Execute the **command** below to verify what version you have:

```
python --version
```

Note: the image here is only of 2.7 but it should be similar if you check for python3

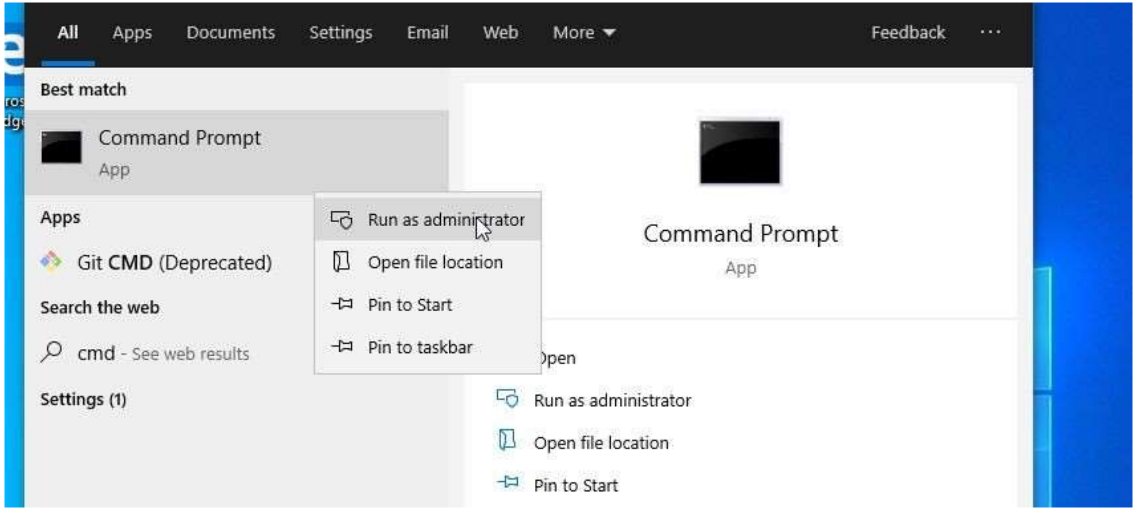
Sometimes your system might have it as

```
python3 --version
```

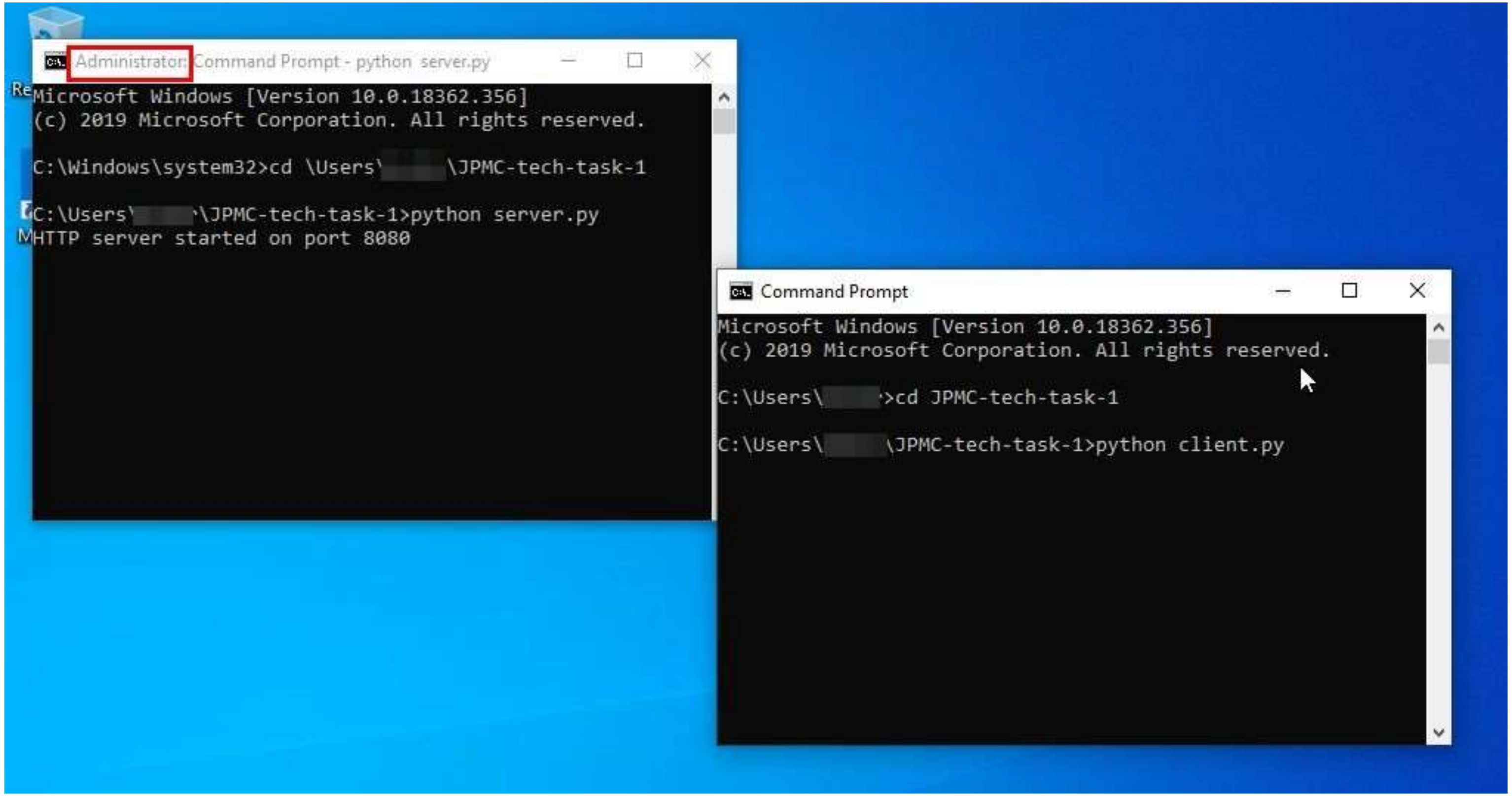
(any python 2.7.x > = 2.7.16 should suffice but the latest 2.7.x is recommended (**2.7.17**)*;*
any python 3.x >= 3.6is fine, latest is recommended (**3.8.0**))

Local Setup (Windows)

- Once you have Python 2.7 or Python3 installed, all you have to do get the application up and running is to start the server and client scripts in two separate cmds (*see image in the next slide*). Ensure that the command line wherein you run `python server.py` is on Administrator mode:



Local Setup (Windows)



Administrator: Command Prompt - python server.py

```
Microsoft Windows [Version 10.0.18362.356]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd \Users\[redacted]\JPMC-tech-task-1

C:\Users\[redacted]\JPMC-tech-task-1>python server.py
HTTP server started on port 8080
```

Command Prompt

```
Microsoft Windows [Version 10.0.18362.356]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\[redacted]>cd JPMC-tech-task-1

C:\Users\[redacted]\JPMC-tech-task-1>python client.py
```

Local Setup (Windows)

- When you open the cmd in admin mode, you'll notice that its current directory starts from C:\Windows\System32. Assuming you cloned the repository in your home directory you need to use the **cd** command to get there. For instance if I cloned it in C:\Users\insidesherpa, then I should do something like:

  cd \Users\insidesherpa\JPMC-tech-task-1
  Or
  cd \Users\insidesherpa\JPMC-tech-task-1-py3

note: the example above isn't what you're going to type on your system. You have a different user account in \Users where you probably cloned the repo. Use that instead...

Local Setup (Windows)

- (*note: just choose to run one server and one client; either the python 2 or python 3 version of server and client applications*. Run the **commands** below on separate command lines, starting with the server and then the client. The commands will vary depending on your primary python version)

  // If python --version = 2.7+, you must be in the JPMC-tech-task-1
  // If python --version = 3+ , you must be in JPMC-tech-task-1-py3 directory
  ```
  python server.py
  python client.py
  ```

  // If your system makes the distinction of python3 as `python3`,
  // you must be in JPMC-tech-task-1-py3 directory
  ```
  python3 server3.py
  python3 client3.py
  ```

If ever you encounter an error when starting the server application, see troubleshooting in this slide

Local Setup: Troubleshooting (Windows)

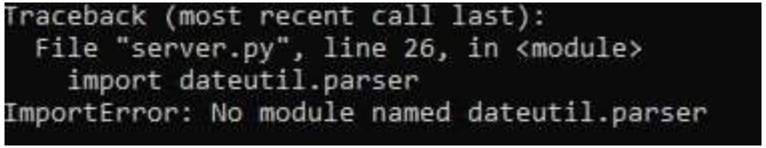
- If you did not encounter any issues, your setup is finished. From here on, you can make changes to the code (see another guide in the module page for this) and eventually arrive at the desired output.

- If you did encounter issues, check if the commonly encountered issues listed in the next few slides will solve your problem:

    - dateutil dependency
    - python not recognized or not returning in your command line
    - Socket unavailable

Local Setup: Troubleshooting (Windows)

- In some cases, dependency issues might arise like when you run `server.py`:



```
Traceback (most recent call last):
  File "server.py", line 26, in <module>
    import dateutil.parser
ImportError: No module named dateutil.parser
```

In this case, you must install pip first. pip is python's package manager for installing python dependencies. Make sure you install pip for the right Python version you're working with in this project. You can check your pip version by **pip --version** and it will tell which python version it maps too

Local Setup: Troubleshooting (Windows)

- Installing pip nowadays usually involves downloading the get-pip.py script. If you followed the instructions in the last slide, it usually involves using the command:

  ```
  curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
  ```

  If you don't have curl, just install it in your system. For mac, it's this way
  Then just run the script using python:

  ```
  //if python --version = 2.7+ this will install pip for python2
  //if python --version = 3+ this will install pip for python3
  python get-pip.py

  //if your system makes the distinction of python3 as `python3` then
  //doing the command below will install pip for python3
  python3 get-pip.py
  ```

Local Setup: Troubleshooting (Windows)

- After that, for pip to be a recognizable command in your terminal/command line, you need to add it in your system's path environment variable
  - On Windows, for Python2.7 it's usually in C:\\Python27\Scripts. It would also be similar for Python3.x if you followed the installation guide for Python earlier (e.g. C:\\Python3X\Scripts)

  - Make sure you open a new command line too and use that instead after doing this

- To edit your system path environment variable it's similar to the slides here.
- Alternatively you can access it doing something like:
  - **C:\\Python27\Scripts\pip.exe <parameters>** (similar for python3x if it was installed in C:\\)
  - <parameters> could be something like **C:\\Python27\Scripts\pip.exe install python-dateutil**
  - Take note though, this assumes that you have your python installed in drive C:\\

Local Setup: Troubleshooting (Windows)

- Then afterwards, you can run the following command on your terminal to install the dependency:

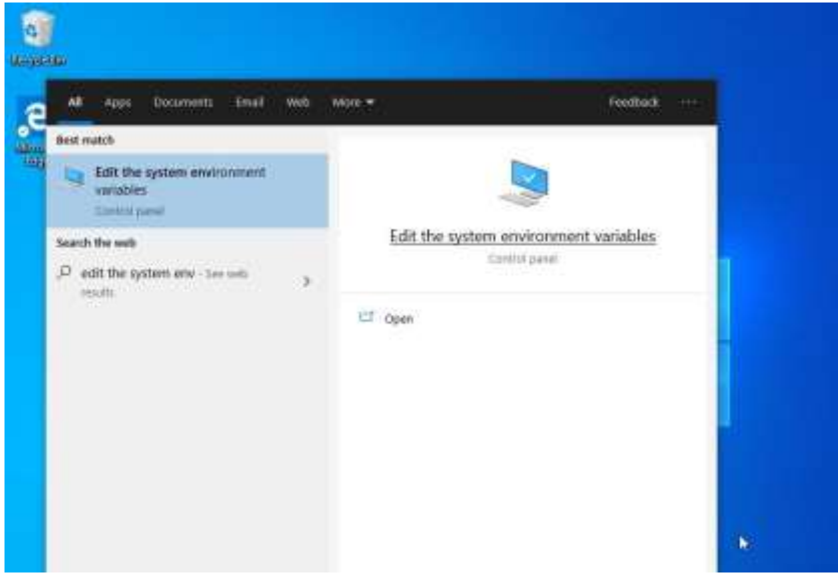
```
pip install python-dateutil
```

Afterwards, you can rerun the server and then rerun the client

Note: For the command above, whatever python version your pip corresponds to (i.e. the output of **pip --version**, that is the python version that will have the dependency installed). So if you're **pip** corresponds to python2.7.x then doing the command above will install python-dateutil for python2.7.x

Local Setup: Troubleshooting (Windows)

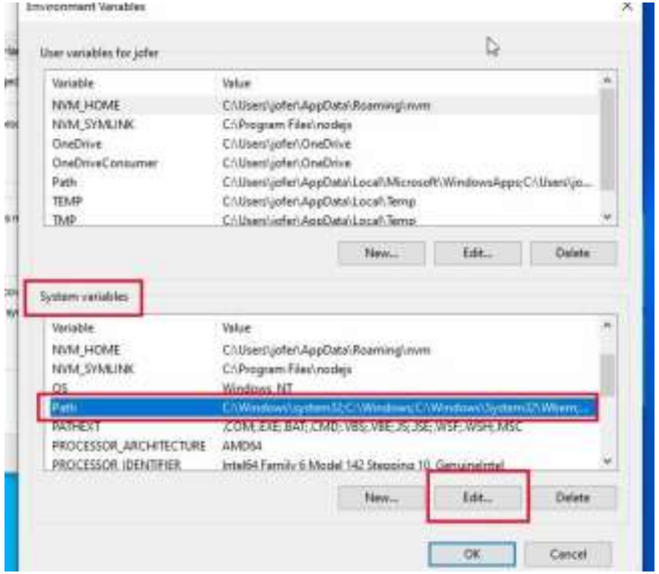
- There are some cases when you tried installing python e.g. through the usual installation process or via other software like anaconda, and when you open your command line and try executing any python command like **python** or **python --version** nothing returns.The problem here is usually because python isn't properly set in your system environment's path variable properly. To do this go to your start menu,search "edit the system environment" and you should be able to see something like the image on the right


All
Apps
Documents
Email
Web
More
Feedback
Best match
Edit the system environment variables
Control panel
Search the web
edit the system env - See web results
Edit the system environment variables
Control panel
Open

Local Setup: Troubleshooting (Windows)

- Click that and click “Environment Variables” in the next window (image on left side)

- You should then come up with the image below:



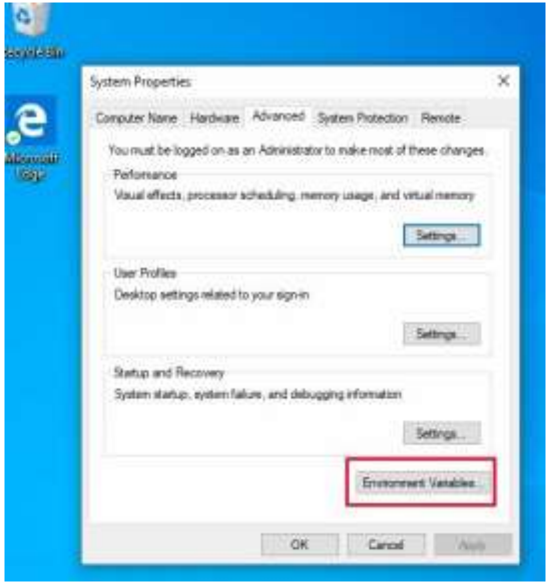
Environment Variables

User variables for jofer

Variable	Value
NVM_HOME	C:\Users\jofer\AppData\Roaming\nvm
NVM_SYMLINK	C:\Program Files\nodejs
OneDrive	C:\Users\jofer\OneDrive
OneDriveConsumer	C:\Users\jofer\OneDrive
Path	C:\Users\jofer\AppData\Local\Microsoft\WindowsApps;C:\Users\jo...
TEMP	C:\Users\jofer\AppData\Local\Temp
TMP	C:\Users\jofer\AppData\Local\Temp

System variables

Variable	Value
NVM_HOME	C:\Users\jofer\AppData\Roaming\nvm
NVM_SYMLINK	C:\Program Files\nodejs
OS	Windows_NT
Path	C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 142 Stepping 10, GenuineIntel



System Properties

You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory

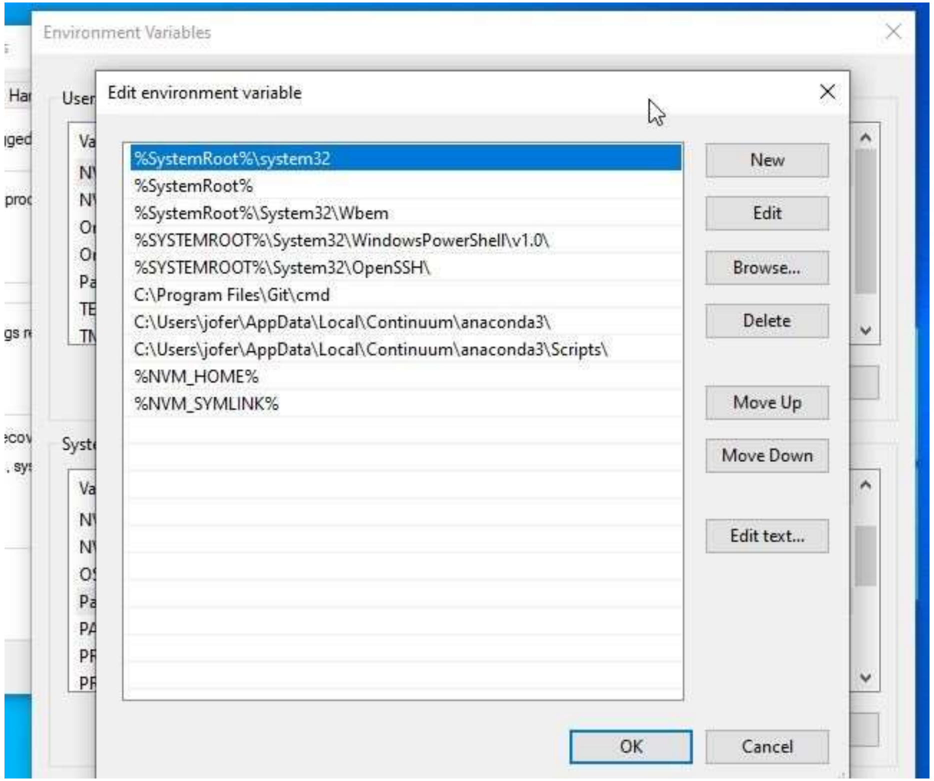
User Profiles
Desktop settings related to your sign-in

Startup and Recovery
System startup, system failure, and debugging information

You want to edit the **Path** under **System variables**

Local Setup: Troubleshooting (Windows)

- You should end up with a window like this after the steps from the previous slide:



```
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\Git\cmd
C:\Users\jofer\AppData\Local\Continuum\anaconda3\
C:\Users\jofer\AppData\Local\Continuum\anaconda3\Scripts\
%NVM_HOME%
%NVM_SYMLINK%
```

You want to make sure the directory where the **Python** executable/application you want to use is in is included in the list of paths.

If you followed how to install **Python2.7.x** in the earlier slides, you should have it in C:\\Python27**.** Make sure to include that in your path.

Same goes for **Python3.x** if you want to enable it. But make sure to remove your Python2.7.x path first .

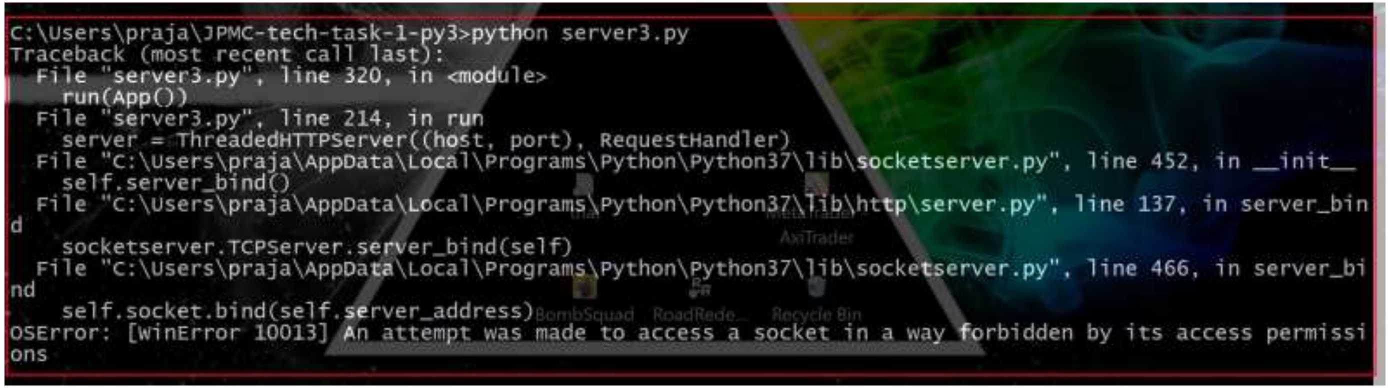
If you installed python using other means, e.g. anaconda, its python executable is located elsewhere but same method of putting the path applies

Don’t forget to restart your command line after setting a new path to reflect the changes...

Local Setup: Troubleshooting (Windows)

- In other cases, you might encounter problems running the server app

CASE A:



```
C:\Users\praja\JPMC-tech-task-1-py3>python server3.py
Traceback (most recent call last):
  File "server3.py", line 320, in <module>
    run(App())
  File "server3.py", line 214, in run
    server = ThreadedHTTPServer((host, port), RequestHandler)
  File "C:\Users\praja\AppData\Local\Programs\Python\Python37\lib\socketserver.py", line 452, in __init__
    self.server_bind()
  File "C:\Users\praja\AppData\Local\Programs\Python\Python37\lib\http\server.py", line 137, in server_bin
d
    socketserver.TCPServer.server_bind(self)
  File "C:\Users\praja\AppData\Local\Programs\Python\Python37\lib\socketserver.py", line 466, in server_bi
nd
    self.socket.bind(self.server_address)
OSError: [WinError 10013] An attempt was made to access a socket in a way forbidden by its access permissi
ons
```

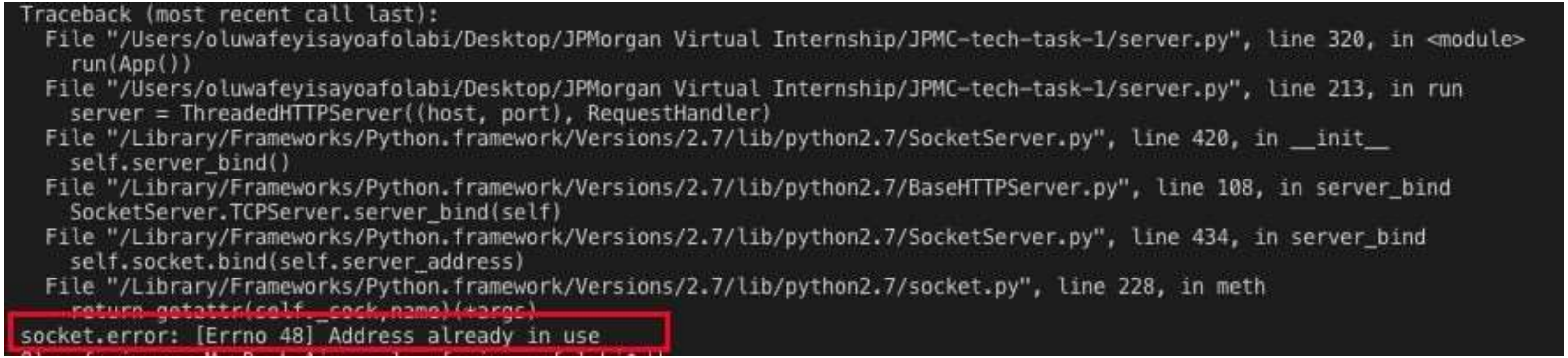
note: the example here is from windows but a similar error might appear for mac

This is most likely because you have a firewall open preventing you from accessing 8080. You can try the following workarounds:

- Temporarily turn off your firewall
- Using any text editor, open the server.py or server3.py in the repository using your code editor and look for the line where it says port = 8080. change that to port = 8085
- Similarly, open the client.py or client3.py and change the line where it has 8080 to 8085

Local Setup: Troubleshooting (Windows)

CASE B:



```
Traceback (most recent call last):
  File "/Users/oluwafeyisayoafolabi/Desktop/JPMorgan Virtual Internship/JPMC-tech-task-1/server.py", line 320, in <module>
    run(App())
  File "/Users/oluwafeyisayoafolabi/Desktop/JPMorgan Virtual Internship/JPMC-tech-task-1/server.py", line 213, in run
    server = ThreadedHTTPServer((host, port), RequestHandler)
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/SocketServer.py", line 420, in __init__
    self.server_bind()
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/BaseHTTPServer.py", line 108, in server_bind
    SocketServer.TCPServer.server_bind(self)
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/SocketServer.py", line 434, in server_bind
    self.socket.bind(self.server_address)
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/socket.py", line 228, in meth
    return getattr(self._sock,name)(*args)
socket.error: [Errno 48] Address already in use
```

In this case make sure you're only running one instance of the server.py because it hooks itself to port 8080, and once that port is used nothing else can use it. If you want to free that up, terminate the old server.py you're running from one of your terminals by hitting cmd+c. Alternatively you can kill the process listening on a port (i.e. in this case 8080) by following this guide

Local Setup: Troubleshooting (Windows)

- If you did encounter any other issues, please post your issue/inquiry here: https://github.com/insidesherpa/JPMC-tech-task-1/issues or https://github.com/insidesherpa/JPMC-tech-task-1-py3/issues depending on what repository you chose to work in. When submitting a query, please don't forget to provide as much context as possible, i.e. your OS, what you've done, what your errors is/are, etc (screenshots would help too)

- You can also submit your query in the module page's support modal that pops out when you click the floating element on the page (see image below)

InsideSherpa

JPMorgan Chase Software Engineering Virtual Experience

Software Engineering Task-1

Module 1 - Interface with a stock price data feed and create tests for it

Disclaimer

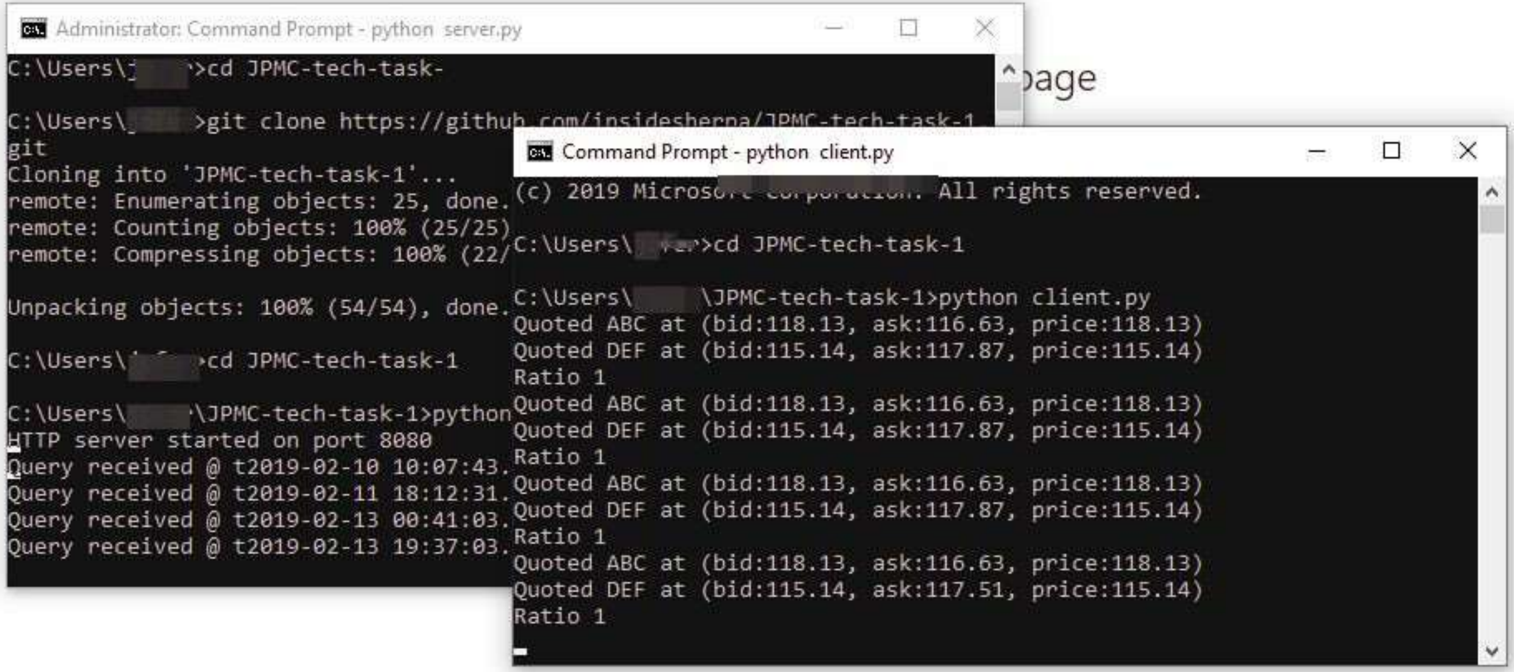
- This guide is only for those who did the setup locally on their machines.

Prerequisite

- Set up should have been done. This means,your server and client applications should have been running with no problems without introducing any changes to the code yet. You can verify this if you get a similar result to any of the following slides that include a picture of the server and client app running together

Prerequisite

Windows OS (left side server, right side client)



Administrator: Command Prompt - python server.py

```
C:\Users\[redacted]>cd JPMC-tech-task-

C:\Users\[redacted]>git clone https://github.com/insidesherpa/JPMC-tech-task-1
git
Cloning into 'JPMC-tech-task-1'...
remote: Enumerating objects: 25, done.
remote: Counting objects: 100% (25/25)
remote: Compressing objects: 100% (22/

Unpacking objects: 100% (54/54), done.

C:\Users\[redacted]>cd JPMC-tech-task-1

C:\Users\[redacted]\JPMC-tech-task-1>python
HTTP server started on port 8080
Query received @ t2019-02-10 10:07:43.
Query received @ t2019-02-11 18:12:31.
Query received @ t2019-02-13 00:41:03.
Query received @ t2019-02-13 19:37:03.
```

Command Prompt - python client.py

```
(c) 2019 Microso[redacted] All rights reserved.

C:\Users\[redacted]>cd JPMC-tech-task-1

C:\Users\[redacted]\JPMC-tech-task-1>python client.py
Quoted ABC at (bid:118.13, ask:116.63, price:118.13)
Quoted DEF at (bid:115.14, ask:117.87, price:115.14)
Ratio 1
Quoted ABC at (bid:118.13, ask:116.63, price:118.13)
Quoted DEF at (bid:115.14, ask:117.87, price:115.14)
Ratio 1
Quoted ABC at (bid:118.13, ask:116.63, price:118.13)
Quoted DEF at (bid:115.14, ask:117.87, price:115.14)
Ratio 1
Quoted ABC at (bid:118.13, ask:116.63, price:118.13)
Quoted DEF at (bid:115.14, ask:117.51, price:115.14)
Ratio 1
```

Objectives

- If you closely inspect the output of the client applications in the previous slides, there are two incorrect things…
    - (1) Ratio is always 1
    - (2) The price of each stock is always the same as its bid_price.
- These are obviously wrong so you job is to fix those things…
- Don’t worry we’ll walk you through how to get these things done

How to make changes to code

- You'll be making changes to the code in the some of the files within the repository you cloned or downloaded to achieve the objectives the task.

- To do this, you can use any text editor your machine has and just open the files in the repository that must be changed (the guide will instruct you in the following slides which files these will be)

- Our recommendation of editors you can use would be VSCode or SublimeText or WebStrom as these are the most commonly used code editors out there.

Making changes in `client.py` (client3.py for python3)

- All the changes you have make to get the right output will be in the client.py file inside the repository (client3.py)

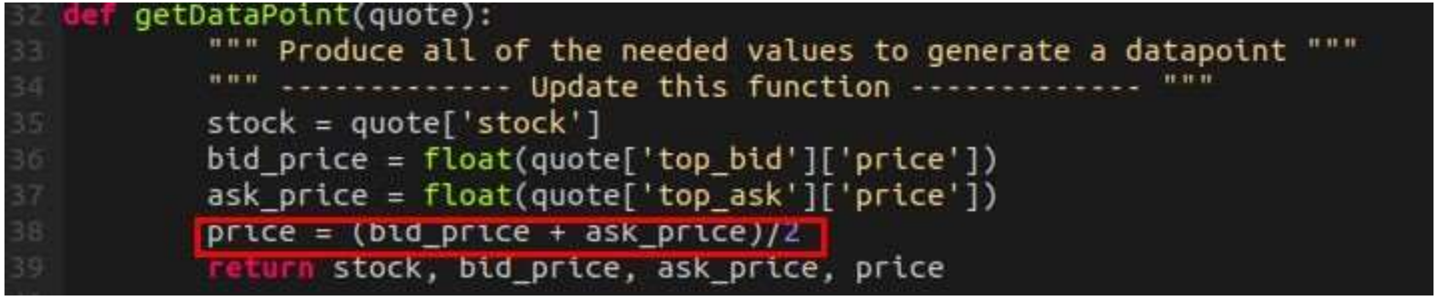
- The changes you need to make will be in the following methods of the file
    - getDataPoint
    - getRatio
    - Main

- The changes for each method will be dissected for each method on the next slide

Making changes in `client.py` (client3.py for python3) **getDataPoint**

getDataPoint. In this method, you'll have to make the modifications to compute for the right stock price. This means you have to change how `price` is computed for. The formula is (`bid_price+ask_price) / 2`.

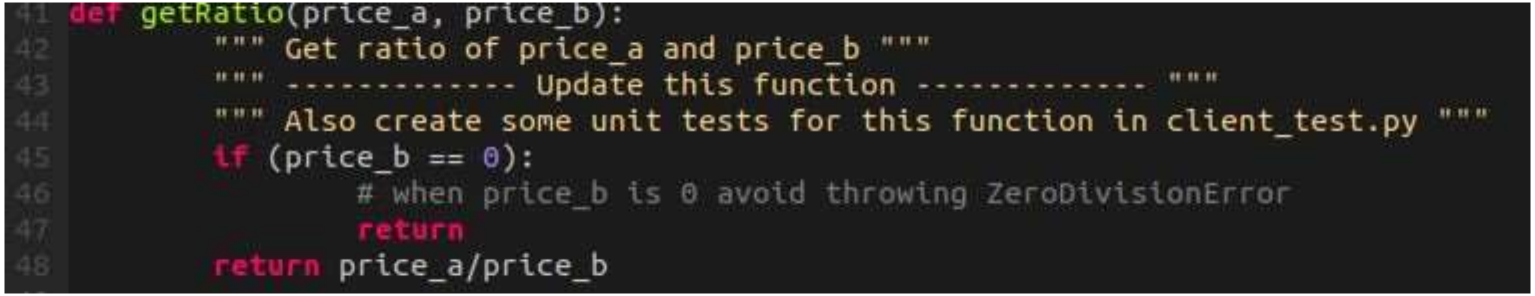
YOU DO NOT NEED TO CHANGE the return value as that is representational of the entire data point. You should end up with something like:



```
def getDataPoint(quote):
        """ Produce all of the needed values to generate a datapoint """
        """ ------------- Update this function ------------- """
        stock = quote['stock']
        bid_price = float(quote['top_bid']['price'])
        ask_price = float(quote['top_ask']['price'])
        price = (bid_price + ask_price)/2
        return stock, bid_price, ask_price, price
```

Making changes in `client.py` (client3.py for python3) **getRatio**

getRatio. In the original case, this method just returns 1 all the time. To correct this, you must change the return value to the ratio of stock **price_a** to stock **price_b**

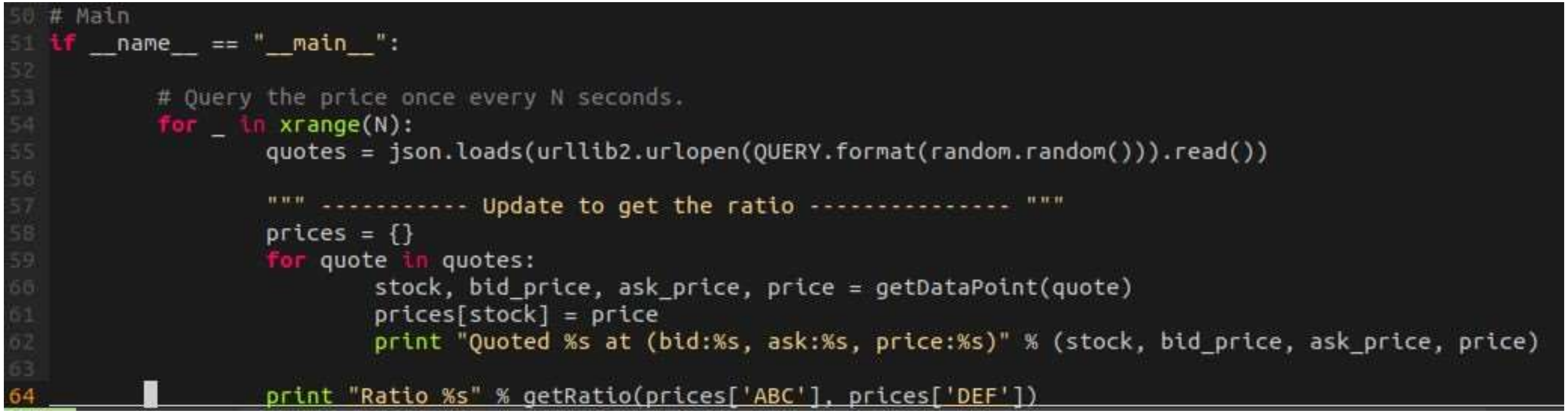


```
def getRatio(price_a, price_b):
        """ Get ratio of price_a and price_b """
        """ ------------- Update this function ------------- """
        """ Also create some unit tests for this function in client_test.py """
        if (price_b == 0):
                # when price_b is 0 avoid throwing ZeroDivisionError
                return
        return price_a/price_b
```

note: that we've also added the condition of the case where in price_b could be zero, i.e. division by zero, in the rare chance that it might happen...

Making changes in `client.py` (client3.py for python3) **main**

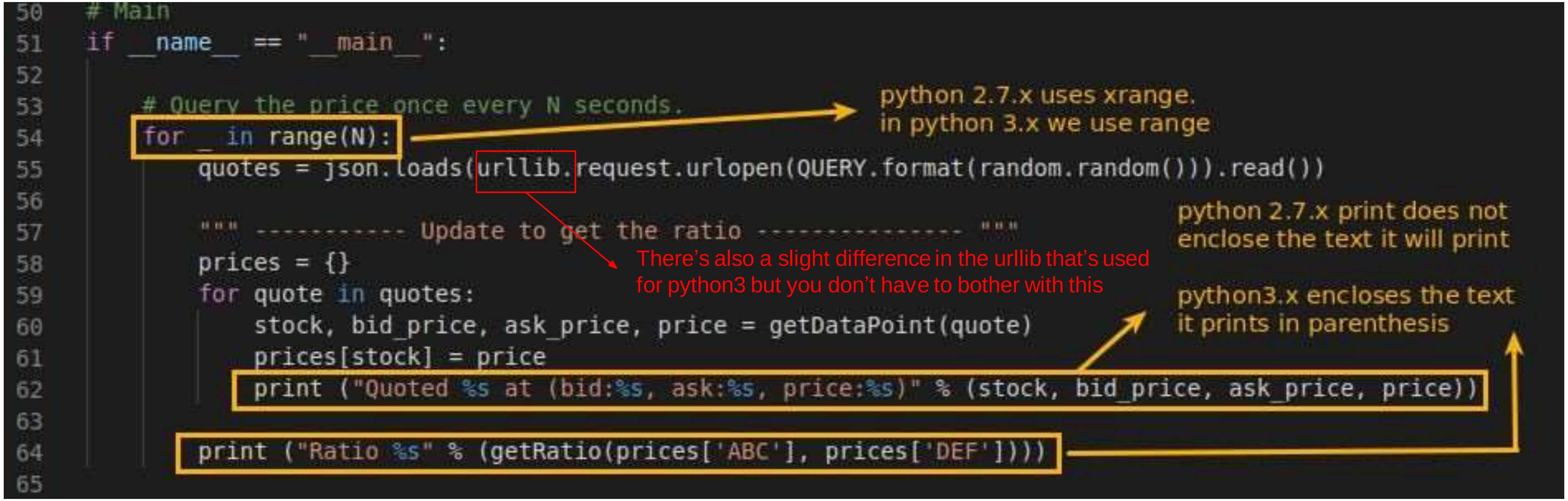
main method. Now that you've fixed the two other methods, it's just a matter of printing the correct values. For every iteration in the main method, you need to store the datapoints you get from getDataPoint method so that you can properly call getRatio and print the right ratio out. (*the below image is for python 2.7.x version*)



```
# Main
if __name__ == "__main__":

        # Query the price once every N seconds.
        for _ in xrange(N):
                quotes = json.loads(urllib2.urlopen(QUERY.format(random.random())).read())

                """ ----------- Update to get the ratio --------------- """
                prices = {}
                for quote in quotes:
                        stock, bid_price, ask_price, price = getDataPoint(quote)
                        prices[stock] = price
                        print "Quoted %s at (bid:%s, ask:%s, price:%s)" % (stock, bid_price, ask_price, price)

                print "Ratio %s" % getRatio(prices['ABC'], prices['DEF'])
```

Making changes in `client.py` (client3.py for python3) **main**

main method. (*the image here is now for python 3.x version*)



```
# Main
if __name__ == "__main__":

    # Query the price once every N seconds.
    for _ in range(N):
        quotes = json.loads(urllib.request.urlopen(QUERY.format(random.random())).read())

        """ ----------- Update to get the ratio --------------- """
        prices = {}
        for quote in quotes:
            stock, bid_price, ask_price, price = getDataPoint(quote)
            prices[stock] = price
            print ("Quoted %s at (bid:%s, ask:%s, price:%s)" % (stock, bid_price, ask_price, price))

        print ("Ratio %s" % (getRatio(prices['ABC'], prices['DEF'])))
```

python 2.7.x uses xrange.
in python 3.x we use range

There's also a slight difference in the urllib that's used for python3 but you don't have to bother with this

python 2.7.x print does not enclose the text it will print

python3.x encloses the text it prints in parenthesis

Making changes in `client.py` (client3.py for python3) **main**

- To review the changes in main (*whether it was in python2 or python3*), what we did was create a **prices** dictionary to store the stock prices. Think of a dictionary as a key-value store wherein you can specify a key and be able to retrieve a value. In our case, they key was the stock name and the value was the price.

- We then used this **prices** dictionary at the end to pass in the right values in the **getRatio** function.

Next step

- You’re done with the main task. You can either create your patch file now by following this guide, or try and do the bonus task first

InsideSherpa

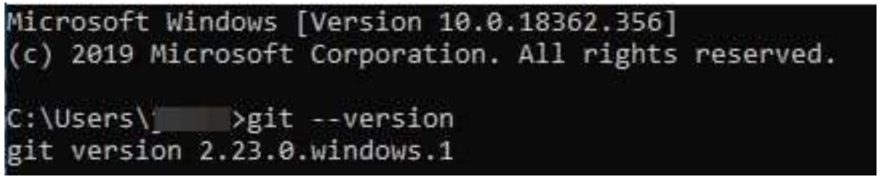
JPMorgan Chase Software Engineering Virtual Experience

Setting up your Windows for the JPMorgan Chase program

Module 2 - Use JPMorgan Chase frameworks and tools

Local Setup (Windows)

- First you must have git installed in your system. Git is usually used by programmers to collaborate with code in a software project. To do install, follow this quick guide. You know you have installed successfully when you get this output on your command line (cmd). (*any git version should suffice but the latest is recommended)*



```
Microsoft Windows [Version 10.0.18362.356]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\[redacted]>git --version
git version 2.23.0.windows.1
```

Local Setup (Windows)

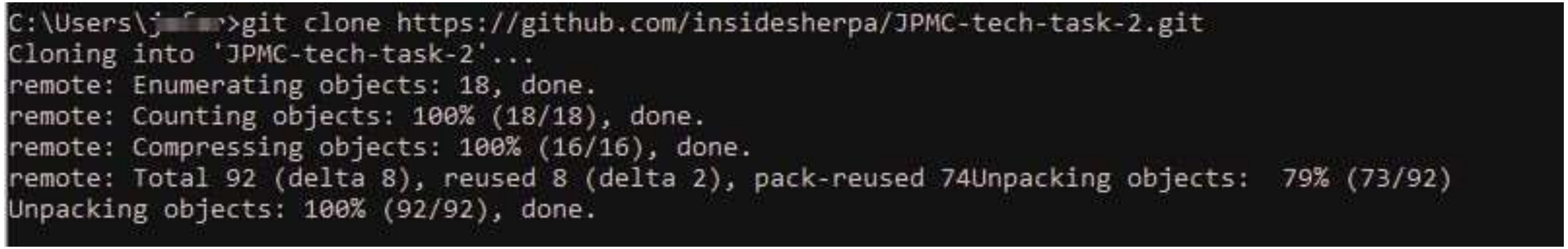
- Once you have git installed, you need a copy of the application code you'll be working with on your machine. To do this, you must execute the following commands on your terminal:

```
git clone https://github.com/insidesherpa/JPMC-tech-task-2.git
git clone https://github.com/insidesherpa/JPMC-tech-task-2-PY3.git
```

- This command will download the code repositories from github to your machine in the current working directory of the terminal you executed the command in. Downloading the 2 repositories above will give you options later

Local Setup (Windows)

- You’ll know you cloned successfully if you have the copy of the application code on your machine:



```
C:\Users\j[redacted]r>git clone https://github.com/insidesherpa/JPMC-tech-task-2.git
Cloning into 'JPMC-tech-task-2'...
remote: Enumerating objects: 18, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 92 (delta 8), reused 8 (delta 2), pack-reused 74Unpacking objects:  79% (73/92)
Unpacking objects: 100% (92/92), done.
```

*note: the image here just does not contain the other repository but it should if you did the previous slides and execute the **dir** command. `dir` just lists the files/folders in the current directory*

*note: take note of the current directory your commandline is in because that is the location where you copied the repository. In the image above it’s in **C:\Users\<username>** (See next slide on how to navigate inside the repository) **AVOID cloning your repository in C:\\Windows\System32***

Local Setup (Windows)

- To access the files inside from the terminal, just change directory by typing:

```
cd JPMC-tech-task-2
```

note: If you choose to work using python3 and your system has version python3.x as default instead of python2.7.x, then choose to go into the other repository you downloaded instead. (otherwise, use the other repo above); ***cd*** *changes the directory your terminal is in. Take note, the* ***cd*** *command assumes the directory you want to go into is inside the current directory you're in. For more info on how to use cd, check here*

```
cd JPMC-tech-task-2-py3
```

Local Setup (Windows)

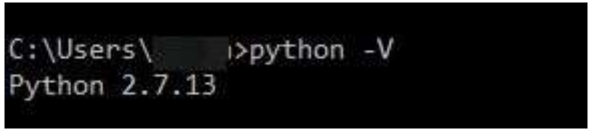
- To clarify, you're only supposed to work on one of the repositories you cloned / downloaded into your system. It all depends on what Python version you primarily use.

- Python is just a scripting / programming language we developers use quite often in the field. This application you'll be working on uses it.

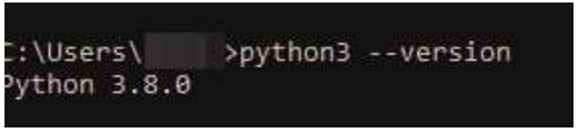
- We'll discuss checking / installing Python in your system in the following slides

Local Setup (Windows)

- Next, you’ll need to have Python 2.7 **and/or** Python 3+ installed on your machine. Follow the instructions here (for python2), and here (for python3) You can verify this on your command line (cmd) if you get a result like:



```
C:\Users\[redacted]>python -V
Python 2.7.13
```



```
:\Users\[redacted]>python3 --version
ython 3.8.0
```

(any python 2.7.x > = 2.7.16 should suffice but the latest 2.7.x is recommended (**2.7.17**)*; any python 3.x >= 3.6is fine, latest is recommended* (**3.8.0**))

Execute the command below to verify what version you have:

```
python --version
```

Note: the image here is only of 2.7 but it should be similar if you check for python3. If you install both versions, make sure the command **python** maps to 2.7.x at least

Sometimes your system might have python3 version mapped to

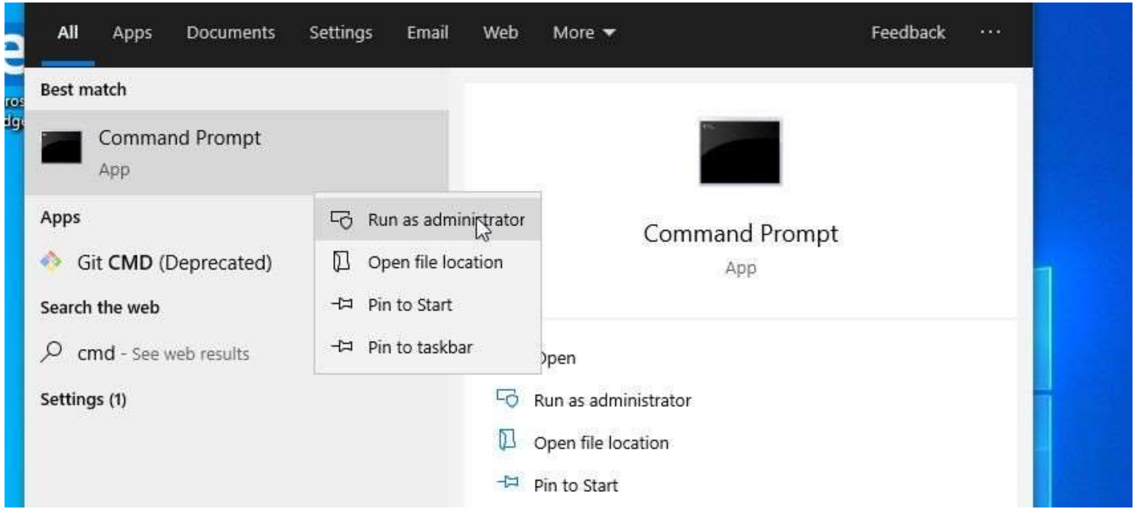
```
python3 --version
```

Local Setup (Windows)

- If something went wrong with the output you got from the previous slide, please check the Troubleshooting slides for Windows in this guide.

Local Setup (Windows)

- Once you have python is installed, all you have to do get the server up and running is to start the server in its own cmd. Ensure that the command line wherein you run the server app is on Administrator mode:



Local Setup (Windows)

- When you open the cmd in admin mode, you'll notice that its current directory starts from C:\Windows\System32. Assuming you cloned it your home directory you need to use the **cd** command to get there. For instance if I cloned it in C:\Users\insidesherpa, then I should do something like:

  cd \Users\insidesherpa\JPMC-tech-task-2
  Or
  cd \Users\insidesherpa\JPMC-tech-task-2-py3

note: the example above isn't what you're going to type on your system. You have a different user account in \Users where you probably cloned the repo. Use that instead...

Local Setup (Windows)

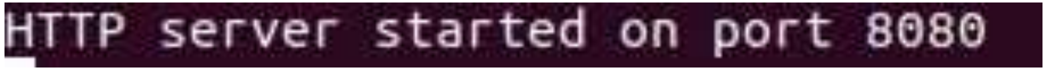
- Once you have Python 2.7 and/or Python 3 installed, you can start the server application in one terminal by just executing it:
  (note: just choose to run one server; either the python 2 or python 3 version of server. Run the commands below depending on your python version)

```
// If python --version = 2.7+, you must be in the JPMC-tech-task-2
// If python --version = 3+ , you must be in JPMC-tech-task-2-py3 directory
python datafeed/server.py

// If your system makes the distinction of python3 as `python3`,
// you must be in JPMC-tech-task-2-py3 directory
python3 datafeed/server3.py
```

If ever you encounter an error when starting the server application, see troubleshooting in this slide

Local Setup (Windows)

- If you've done the previous slide, then you should get something similar to the pic below when you ran the server.



```
HTTP server started on port 8080
```

- To be clear, the server application isn't stuck. It's behaving perfectly normal here. The reason why it's just like that for now is because it's just listening for requests

- For us to be able to make requests, we have to start the client application. The following slides will help you do that.

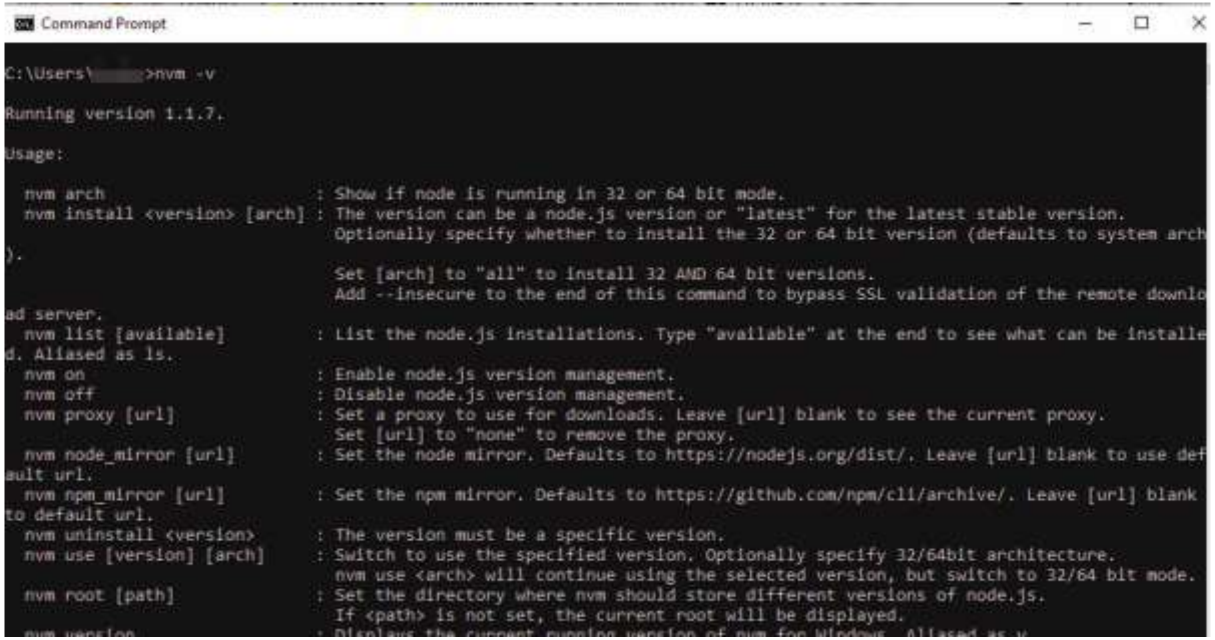
Local Setup (Windows)

- In a separate command line, let's install the other remaining dependencies i.e. Node and Npm. Node is a JavaScript runtime environment that executes JavaScript code outside of a browser and Npm is node's package manager that helps us to install other libraries/dependencies we'll be using in our web application app.

- To install node and npm and be able to manage versions seamlessly we will install NVM (node version manager). Once this is on your machine you can basically install any version of node and npm and switch depending on a project's needs. Follow these instructions to install nvm for windows.

Local Setup (Windows)

- You will know you've successfully installed nvm if you get a similar result below when you type the command **nvm -v**



Command Prompt

```
C:\Users\[redacted]>nvm -v

Running version 1.1.7.

Usage:

  nvm arch                     : Show if node is running in 32 or 64 bit mode.
  nvm install <version> [arch] : The version can be a node.js version or "latest" for the latest stable version.
                                 Optionally specify whether to install the 32 or 64 bit version (defaults to system arch
).
                                 Set [arch] to "all" to install 32 AND 64 bit versions.
                                 Add --insecure to the end of this command to bypass SSL validation of the remote downlo
ad server.
  nvm list [available]         : List the node.js installations. Type "available" at the end to see what can be installe
d. Aliased as ls.
  nvm on                       : Enable node.js version management.
  nvm off                      : Disable node.js version management.
  nvm proxy [url]              : Set a proxy to use for downloads. Leave [url] blank to see the current proxy.
                                 Set [url] to "none" to remove the proxy.
  nvm node_mirror [url]        : Set the node mirror. Defaults to https://nodejs.org/dist/. Leave [url] blank to use def
ault url.
  nvm npm_mirror [url]         : Set the npm mirror. Defaults to https://github.com/npm/cli/archive/. Leave [url] blank
to default url.
  nvm uninstall <version>      : The version must be a specific version.
  nvm use [version] [arch]     : Switch to use the specified version. Optionally specify 32/64bit architecture.
                                 nvm use <arch> will continue using the selected version, but switch to 32/64 bit mode.
  nvm root [path]              : Set the directory where nvm should store different versions of node.js.
                                 If <path> is not set, the current root will be displayed.
  nvm version                  : Displays the current running version of nvm for Windows. Aliased as v.
```

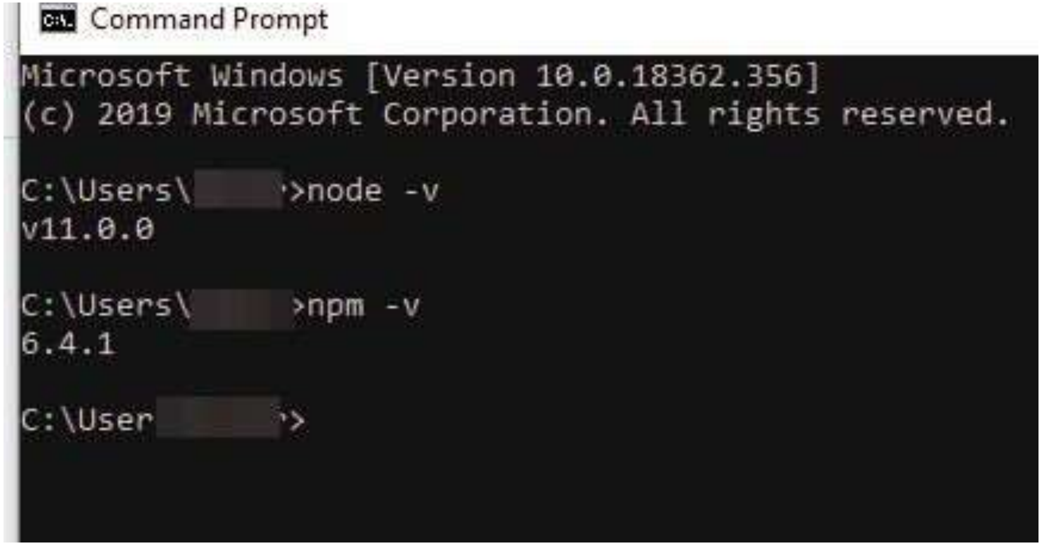
Local Setup (Windows)

- Now, we just need to install the right node version using nvm and that would consequently get us the right npm version as well. We do this by executing the commands:

```
nvm install v11.0.0

nvm use 11.0.0
```

- You should end up with a similar result in the next slide after doing the commands above

Local Setup (Windows)


Command Prompt

```
Microsoft Windows [Version 10.0.18362.356]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\[redacted]>node -v
v11.0.0

C:\Users\[redacted]>npm -v
6.4.1

C:\User[redacted]>
```

To check your node version type and enter:

node -v

To check your npm version type and enter:

npm -v

<u>Take note:</u> If you open a new terminal after all this, make sure to recheck your node version and npm version. It might be the case you've switched to a different version so just execute `nvm use v11.0.0` again if ever...

Local Setup (Windows)

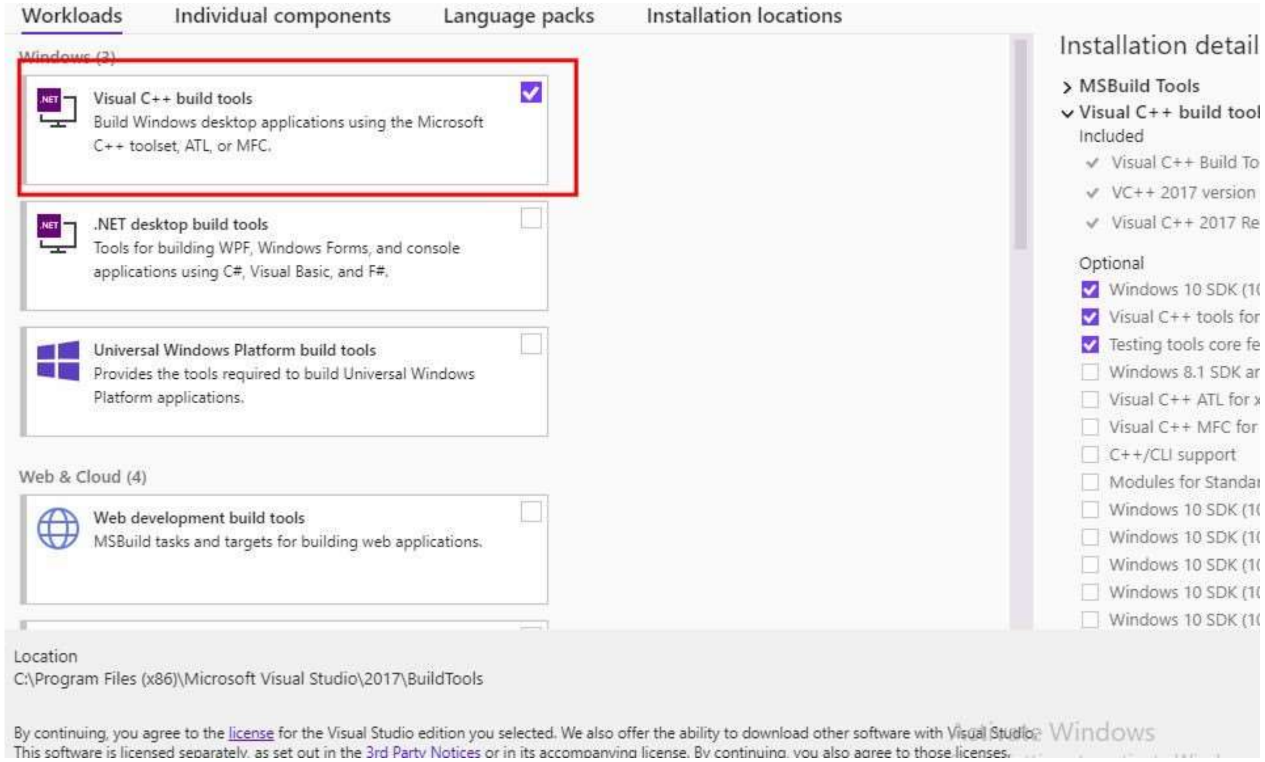
- Because windows' nvm isn't exactly like mac's or linux's, there's still a couple more dependencies we have to install in order to get this whole application to work.

- We have to install Visual C++ Build Environment via Visual Studio Build Tools. Run the downloaded .exe file and make sure to have the basics installed i.e "Visual C++ Build Tools on your machine. This is actually need because of node-gyp. After installing, make sure to run in your command line:

```
npm config set msvs_version 2017
```

(*see image in the next slide*)

Local Setup (Windows)



Workloads | Individual components | Language packs | Installation locations

Windows (3)

- Visual C++ build tools [✓]
  Build Windows desktop applications using the Microsoft C++ toolset, ATL, or MFC.
- .NET desktop build tools
  Tools for building WPF, Windows Forms, and console applications using C#, Visual Basic, and F#.
- Universal Windows Platform build tools
  Provides the tools required to build Universal Windows Platform applications.

Web & Cloud (4)

- Web development build tools
  MSBuild tasks and targets for building web applications.

Installation detail

> MSBuild Tools

∨ Visual C++ build tool

Included

- ✓ Visual C++ Build To
- ✓ VC++ 2017 version
- ✓ Visual C++ 2017 Re

Optional

- [✓] Windows 10 SDK (1(
- [✓] Visual C++ tools for
- [✓] Testing tools core fe
- [ ] Windows 8.1 SDK ar
- [ ] Visual C++ ATL for x
- [ ] Visual C++ MFC for
- [ ] C++/CLI support
- [ ] Modules for Standa
- [ ] Windows 10 SDK (1(
- [ ] Windows 10 SDK (1(
- [ ] Windows 10 SDK (1(
- [ ] Windows 10 SDK (1(
- [ ] Windows 10 SDK (1(

Location
C:\Program Files (x86)\Microsoft Visual Studio\2017\BuildTools

By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

This is how installing visual C++ build environment would look like when you run the .exe file

Local Setup (Windows)

- Finally, to start the client application, all we have to do would be to run the commands below

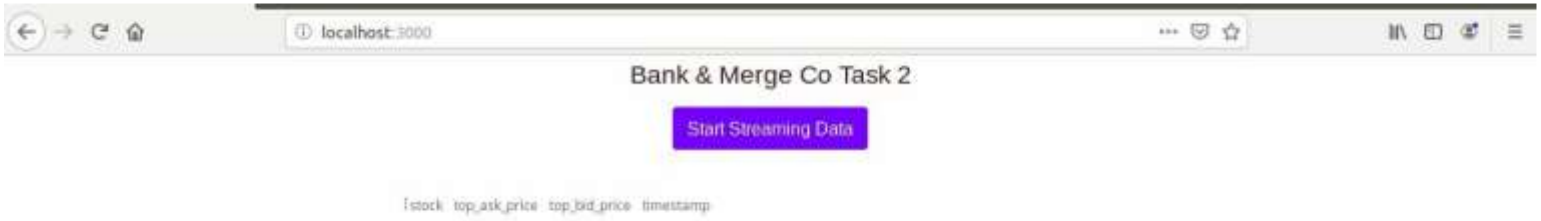
```
npm install
```

npm start

<u>Note:</u> If **npm install** succeeded for you (i.e. no errors) you don't have to do it again… But it always has to be successful first before you execute **npm start**

- If all goes well , you should end up with a similar result in the next slide. If you you encounter problems with npm install, particularly relating to node-gyp try the other windows options of installing here. If there are other issues, please refer to the Troubleshooting slides in this guide.

Local Setup (Windows)



Note: This part assumes no errors came out of **npm install.** Some data should show if you click on the “Start Streaming Data” button. If nothing is showing, check the command line where you’re running the server and see if anything printed out, like “Query received”. If there’s something like that and you’re not seeing results, then try using a different browser (e.g. Chrome/Firefox) and checking localhost:3000. If no response like “Query received” got printed in the command line running the server, you might be experiencing this issue.

Local Setup: Troubleshooting (Windows)

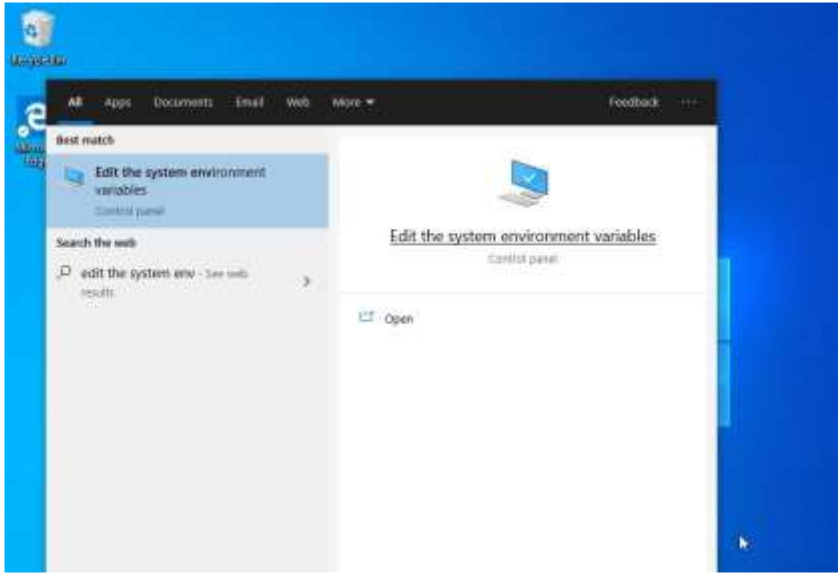
- If you did not encounter any issues, your setup is finished. From here on, you can make changes to the code and eventually arrive at the desired output.

- If you did encounter issues, check if the commonly encountered issues listed in the next few slides will solve your problem:

    - python not recognized or not returning in your command line
    - dateutil dependency
    - npm install or npm start errors
    - Socket unavailable

Local Setup: Troubleshooting (Windows)

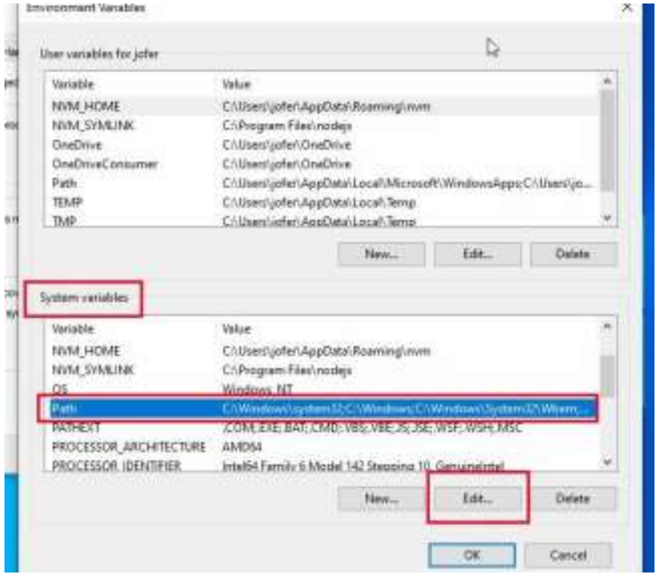
- There are some cases when you tried installing python e.g. through the usual installation process or via other software like anaconda, and when you open your command line and try executing any python command like **python** or **python --version** nothing returns.The problem here is usually because python isn't properly set in your system environment's path variable properly. To do this go to your start menu,search "edit the system environment" and you should be able to see something like the image on the right


All
Apps
Documents
Email
Web
More
Feedback
Best match
Edit the system environment variables
Control panel
Search the web
edit the system env - See web results
Edit the system environment variables
Control panel
Open

Local Setup: Troubleshooting (Windows)

- Click that and click “Environment Variables” in the next window (image on left side)

- You should then come up with the image below:



Environment Variables

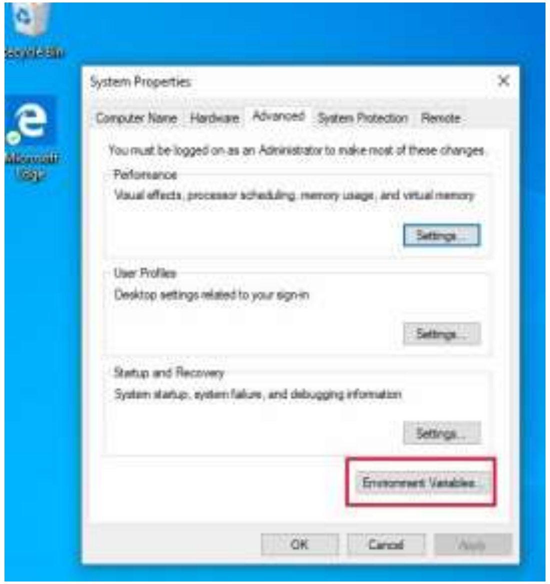
User variables for jofer

Variable	Value
NVM_HOME	C:\Users\jofer\AppData\Roaming\nvm
NVM_SYMLINK	C:\Program Files\nodejs
OneDrive	C:\Users\jofer\OneDrive
OneDriveConsumer	C:\Users\jofer\OneDrive
Path	C:\Users\jofer\AppData\Local\Microsoft\WindowsApps;C:\Users\jo...
TEMP	C:\Users\jofer\AppData\Local\Temp
TMP	C:\Users\jofer\AppData\Local\Temp

System variables

Variable	Value
NVM_HOME	C:\Users\jofer\AppData\Roaming\nvm
NVM_SYMLINK	C:\Program Files\nodejs
OS	Windows_NT
Path	C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 142 Stepping 10, GenuineIntel

You want to edit the **Path** under **System variables**



System Properties

Computer Name | Hardware | Advanced | System Protection | Remote

You must be logged on as an Administrator to make most of these changes.

Performance

Visual effects, processor scheduling, memory usage, and virtual memory

User Profiles

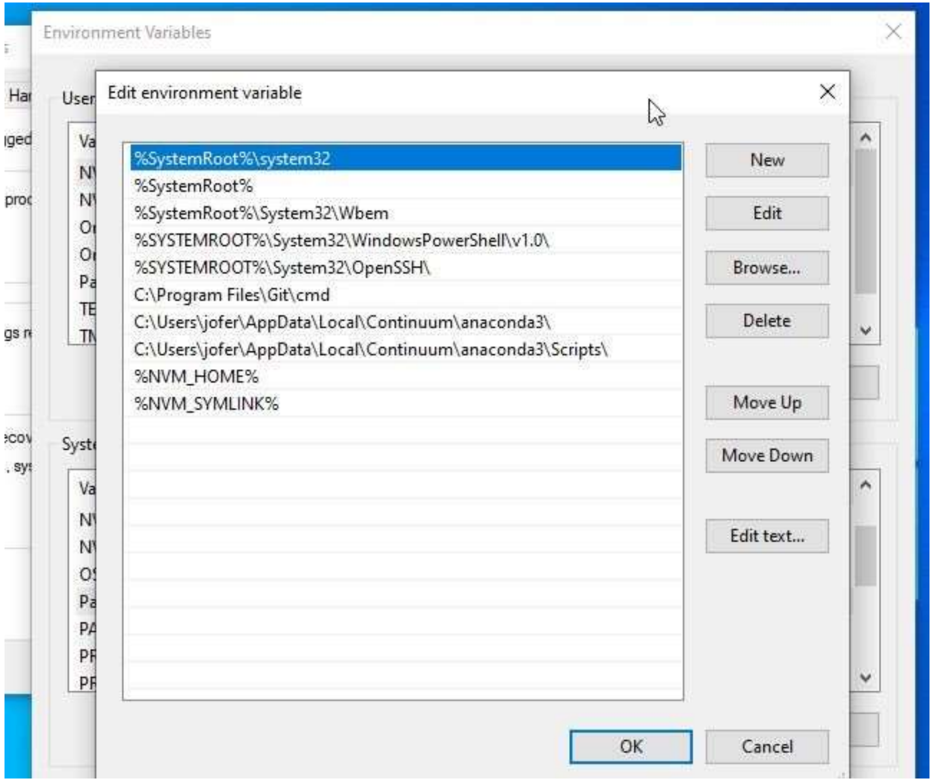
Desktop settings related to your sign-in

Startup and Recovery

System startup, system failure, and debugging information

Local Setup: Troubleshooting (Windows)

- You should end up with a window like this after the steps from the previous slide:



```
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\Git\cmd
C:\Users\jofer\AppData\Local\Continuum\anaconda3\
C:\Users\jofer\AppData\Local\Continuum\anaconda3\Scripts\
%NVM_HOME%
%NVM_SYMLINK%
```

You want to make sure the directory where the **Python** executable/application you want to use is in is included in the list of paths.

If you followed how to install **Python2.7.x** in the earlier slides, you should have it in C:\\Python27**.** Make sure to include that in your path.

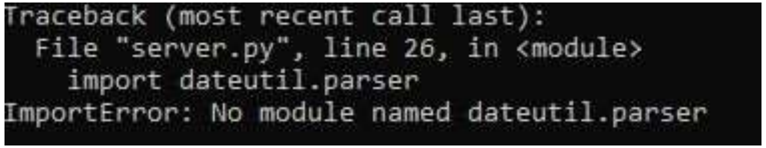
Same goes for **Python3.x** if you want to enable it. But make sure to remove your Python2.7.x path first .

If you installed python using other means, e.g. anaconda, its python executable is located elsewhere but same method of putting the path applies

Don’t forget to restart your command line after setting a new path to reflect the changes...

Local Setup: Troubleshooting (Windows)

- In some cases, dependency issues might arise like when you run `server.py`:



```
Traceback (most recent call last):
  File "server.py", line 26, in <module>
    import dateutil.parser
ImportError: No module named dateutil.parser
```

In this case, you must install pip first. pip is python’s package manager for installing python dependencies. Make sure you install pip for the right Python version you’re working with in this project. You can check your pip version by **pip --version** and it will tell which python version it maps too

Local Setup: Troubleshooting (Windows)

- Installing pip nowadays usually involves downloading the get-pip.py script. If you followed the instructions in the last slide, it usually involves using the command:

  ```
  curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
  ```

  If you don't have curl, just copy a python file, rename it to get-pip.py and replace all the contents of the python file to what's in get-pip.py

  ```
  //if python --version = 2.7+ this will install pip for python2
  //if python --version = 3+ this will install pip for python3
  python get-pip.py

  //if your system makes the distinction of python3 as `python3` then
  //doing the command below will install pip for python3
  python3 get-pip.py
  ```

Local Setup: Troubleshooting (Windows)

- After that, for pip to be a recognizable command in your terminal/command line, you need to add it in your system's path environment variable
  - On Windows, for Python2.7 it's usually in C:\\Python27\Scripts. It would also be similar for Python3.x if you followed the installation guide for Python earlier (e.g. C:\\Python3X\Scripts)
  - Make sure you open a new command line too and use that instead after doing this
- To edit your system path environment variable it's similar to the slides here.
- Alternatively you can access it doing something like:
  - **C:\\Python27\Scripts\pip.exe <parameters>** (similar for python3x if it was installed in C:\\)
  - <parameters> could be something like **C:\\Python27\Scripts\pip.exe install python-dateutil**
  - Take note though, this assumes that you have your python installed in drive C:\\

Local Setup: Troubleshooting (Windows)

- Then afterwards, you can run the following command on your terminal to install the dependency:

```
pip install python-dateutil
```

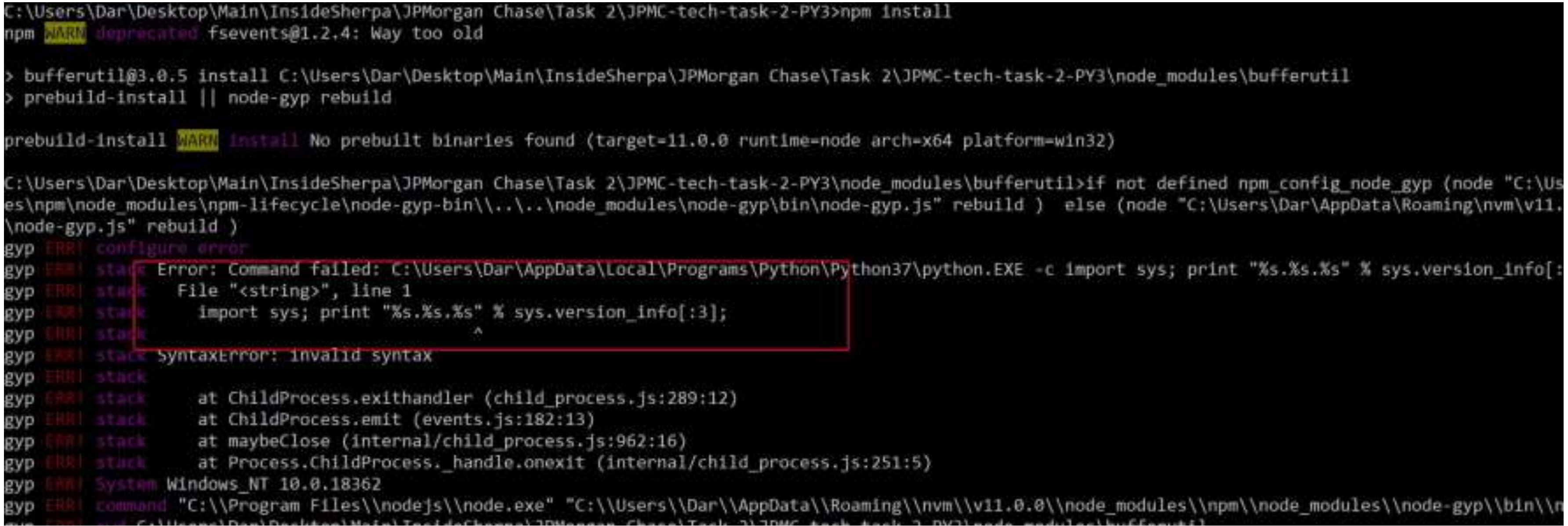
Afterwards, you can rerun the server and then rerun the client

Note: For the command above, whatever python version your pip corresponds to (i.e. the output of **pip --version**, that is the python version that will have the dependency installed). So if you're **pip** corresponds to python2.7.x then doing the command above will install python-dateutil for python2.7.x

Local Setup: Troubleshooting (Windows)

- In other cases, you might encounter problems after doing **npm install**. Errors like the following might appear on your end

CASE A:



```
C:\Users\Dar\Desktop\Main\InsideSherpa\JPMorgan Chase\Task 2\JPMC-tech-task-2-PY3>npm install
npm WARN deprecated fsevents@1.2.4: Way too old

> bufferutil@3.0.5 install C:\Users\Dar\Desktop\Main\InsideSherpa\JPMorgan Chase\Task 2\JPMC-tech-task-2-PY3\node_modules\bufferutil
> prebuild-install || node-gyp rebuild

prebuild-install WARN install No prebuilt binaries found (target=11.0.0 runtime=node arch=x64 platform=win32)

C:\Users\Dar\Desktop\Main\InsideSherpa\JPMorgan Chase\Task 2\JPMC-tech-task-2-PY3\node_modules\bufferutil>if not defined npm_config_node_gyp (node "C:\Us
es\npm\node_modules\npm-lifecycle\node-gyp-bin\\..\..\node_modules\node-gyp\bin\node-gyp.js" rebuild )  else (node "C:\Users\Dar\AppData\Roaming\nvm\v11.
\node-gyp.js" rebuild )
gyp ERR! configure error
gyp ERR! stack Error: Command failed: C:\Users\Dar\AppData\Local\Programs\Python\Python37\python.EXE -c import sys; print "%s.%s.%s" % sys.version_info[:
gyp ERR! stack   File "<string>", line 1
gyp ERR! stack     import sys; print "%s.%s.%s" % sys.version_info[:3];
gyp ERR! stack                                   ^
gyp ERR! stack SyntaxError: invalid syntax
gyp ERR! stack
gyp ERR! stack     at ChildProcess.exithandler (child_process.js:289:12)
gyp ERR! stack     at ChildProcess.emit (events.js:182:13)
gyp ERR! stack     at maybeClose (internal/child_process.js:962:16)
gyp ERR! stack     at Process.ChildProcess._handle.onexit (internal/child_process.js:251:5)
gyp ERR! System Windows_NT 10.0.18362
gyp ERR! command "C:\\Program Files\\nodejs\\node.exe" "C:\\Users\\Dar\\AppData\\Roaming\\nvm\\v11.0.0\\node_modules\\npm\\node_modules\\node-gyp\\bin\\n
```

Local Setup: Troubleshooting (Windows)

- **CASE A**: If you're having a problem similar to this situation, most likely, you haven't set **python** in npm to map to **python2.7.x**. Make sure you've installed **python 2.7.x** first and make sure you map the **python** command npm uses to **python 2.7.x** . Here's a reference on how to do it.

- In our setup node-gyp is called by way of npm, so in order for you to avoid the error, you have to explicitly tell npm what python version to use. Thus, you have to set npm's 'python' config key to the appropriate value:

  - **npm config set python /path/to/executable/python** where /path/to/executable/python is the directory where the python2.7.x binary is, for example in /usr/bin/python usually in linux/mac or C:\\Python27 for Windows

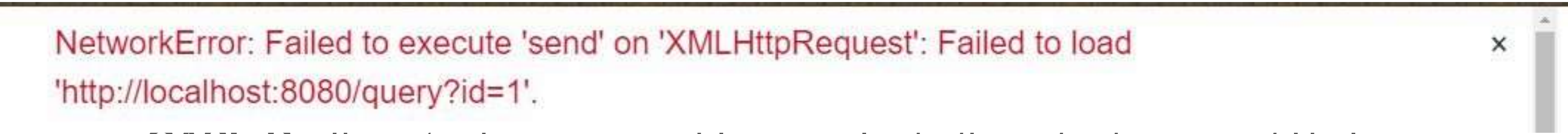
Local Setup: Troubleshooting (Windows)

- **CASE A (continuation)**: Alternatively, you can set **python** in your system to map to **python2.7.x**. Make sure you've installed **python 2.7.x** first and make sure you map the **python** command to **python 2.7.x** (meaning **python --version** should output 2.7.x) Check the slides on how to set this via the system environment path variable in the earlier slides.

- It could help that you also temporarily remove in that same path variable the mapping to **python3.x** if you also have it in your system. Just place it back after you **npm install** successfully and if you plan on using the **py3** version of the repository...

Local Setup: Troubleshooting (Windows)

CASE B:

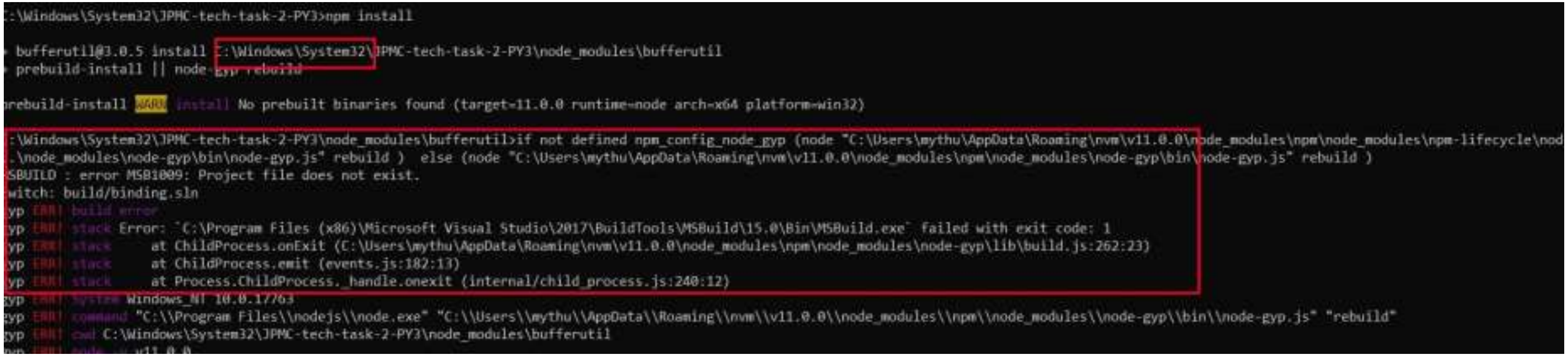

NetworkError: Failed to execute 'send' on 'XMLHttpRequest': Failed to load 'http://localhost:8080/query?id=1'.

- **CASE B:** If you´re having a problem similar to this situation, most likely, you aren't running the server application alongside the client app. Make sure you have another terminal open wherein you're running the server app which is run via **python datafeed/server.py** or **python datafeed/server3.py**

  If running the server app errors out for you might be experiencing this other problem (see linked slide)

Local Setup: Troubleshooting (Windows)

CASE C:



```
:\Windows\System32\JPMC-tech-task-2-PY3>npm install

 bufferutil@3.0.5 install C:\Windows\System32\JPMC-tech-task-2-PY3\node_modules\bufferutil
 prebuild-install || node-gyp rebuild

rebuild-install WARN install No prebuilt binaries found (target=11.0.0 runtime=node arch=x64 platform=win32)

:\Windows\System32\JPMC-tech-task-2-PY3\node_modules\bufferutil>if not defined npm_config_node_gyp (node "C:\Users\mythu\AppData\Roaming\nvm\v11.0.0\node_modules\npm\node_modules\npm-lifecycle\nod
.\node_modules\node-gyp\bin\node-gyp.js" rebuild )  else (node "C:\Users\mythu\AppData\Roaming\nvm\v11.0.0\node_modules\npm\node_modules\node-gyp\bin\node-gyp.js" rebuild )
SBUILD : error MSB1009: Project file does not exist.
witch: build/binding.sln
yp ERR! build error
yp ERR! stack Error: `C:\Program Files (x86)\Microsoft Visual Studio\2017\BuildTools\MSBuild\15.0\Bin\MSBuild.exe` failed with exit code: 1
yp ERR! stack     at ChildProcess.onExit (C:\Users\mythu\AppData\Roaming\nvm\v11.0.0\node_modules\npm\node_modules\node-gyp\lib\build.js:262:23)
yp ERR! stack     at ChildProcess.emit (events.js:182:13)
yp ERR! stack     at Process.ChildProcess._handle.onexit (internal/child_process.js:240:12)
gyp ERR! System Windows_NT 10.0.17763
gyp ERR! command "C:\\Program Files\\nodejs\\node.exe" "C:\\Users\\mythu\\AppData\\Roaming\\nvm\\v11.0.0\\node_modules\\npm\\node_modules\\node-gyp\\bin\\node-gyp.js" "rebuild"
gyp ERR! cwd C:\Windows\System32\JPMC-tech-task-2-PY3\node_modules\bufferutil
```

Case C: This problem is most likely due to the fact that you're in C:\\Windows\System32. Do not clone the repo here as mentioned. Clone in C:\\Users\<some_account>\ instead.

Local Setup: Troubleshooting (Windows)

- After trying the earlier suggestions for troubleshooting and **npm install** still errors out for you, try downloading the **node_modules** here.

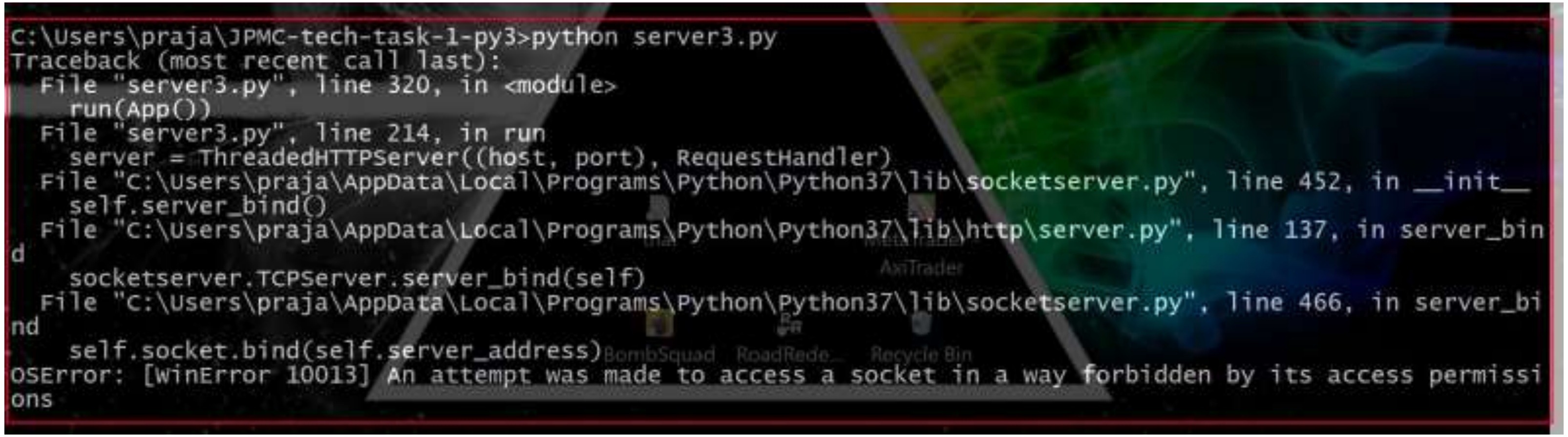
- Then replace the node_modules inside your copy of the repo with the one you downloaded *(make sure the folder is still named node_modules in your repo)*

- Afterwards, you can go ahead and just execute **npm start** (no need to run npm install)

Local Setup: Troubleshooting (Windows)

- In other cases, you might encounter problems running the server app

CASE A:



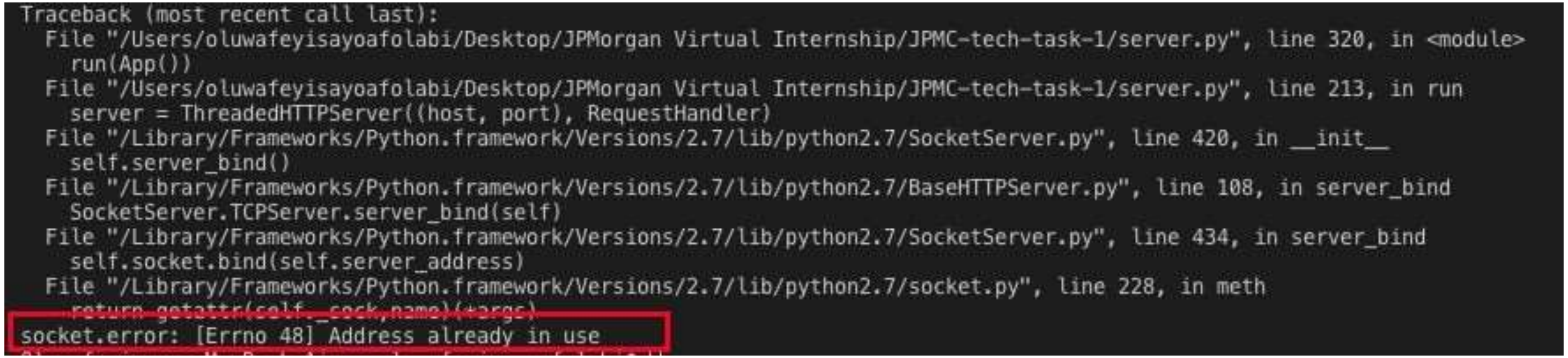
```
C:\Users\praja\JPMC-tech-task-1-py3>python server3.py
Traceback (most recent call last):
  File "server3.py", line 320, in <module>
    run(App())
  File "server3.py", line 214, in run
    server = ThreadedHTTPServer((host, port), RequestHandler)
  File "C:\Users\praja\AppData\Local\Programs\Python\Python37\lib\socketserver.py", line 452, in __init__
    self.server_bind()
  File "C:\Users\praja\AppData\Local\Programs\Python\Python37\lib\http\server.py", line 137, in server_bin
d
    socketserver.TCPServer.server_bind(self)
  File "C:\Users\praja\AppData\Local\Programs\Python\Python37\lib\socketserver.py", line 466, in server_bi
nd
    self.socket.bind(self.server_address)
OSError: [WinError 10013] An attempt was made to access a socket in a way forbidden by its access permissi
ons
```

This is most likely because you have a firewall open preventing you from accessing 8080. You can try the following workarounds:

- Temporarily turn off your firewall
- Using any text editor, open the datafeed/server.py in the repository using your code editor and look for the line where it says port = 8080. change that to port = 8085
- Similarly, open the src/DataStreamer.ts and change the line where it has 8080 to 8085

Local Setup: Troubleshooting (Windows)

CASE B:



```
Traceback (most recent call last):
  File "/Users/oluwafeyisayoafolabi/Desktop/JPMorgan Virtual Internship/JPMC-tech-task-1/server.py", line 320, in <module>
    run(App())
  File "/Users/oluwafeyisayoafolabi/Desktop/JPMorgan Virtual Internship/JPMC-tech-task-1/server.py", line 213, in run
    server = ThreadedHTTPServer((host, port), RequestHandler)
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/SocketServer.py", line 420, in __init__
    self.server_bind()
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/BaseHTTPServer.py", line 108, in server_bind
    SocketServer.TCPServer.server_bind(self)
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/SocketServer.py", line 434, in server_bind
    self.socket.bind(self.server_address)
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/socket.py", line 228, in meth
    return getattr(self._sock,name)(*args)
socket.error: [Errno 48] Address already in use
```

In this case make sure you're only running one instance of the server.py because it hooks itself to port 8080, and once that port is used nothing else can use it. If you want to free that up, terminate the old server.py you're running from one of your terminals by hitting ctrl+c. Alternatively you can kill the process listening on a port (i.e. in this case 8080) by following this guide

Local Setup: Troubleshooting (Windows)

- If you did encounter any other issues not mentioned here, please post your issue/inquiry here: https://github.com/insidesherpa/JPMC-tech-task-2/issues or https://github.com/insidesherpa/JPMC-tech-task-2-py3/issues depending on what repository you chose to work in. When submitting a query, please don't forget to provide as much context as possible, i.e. your OS, what you've done, what your errors is/are, etc (screenshots would help too)

- You can also submit your query in the module page's support modal that pops out when you click the floating element on the page (see image below)

InsideSherpa

JPMorgan Chase Software Engineering Virtual Experience

Software Engineering Task-2

Module 2 - Use JPMorgan Chase frameworks and tools

Disclaimer

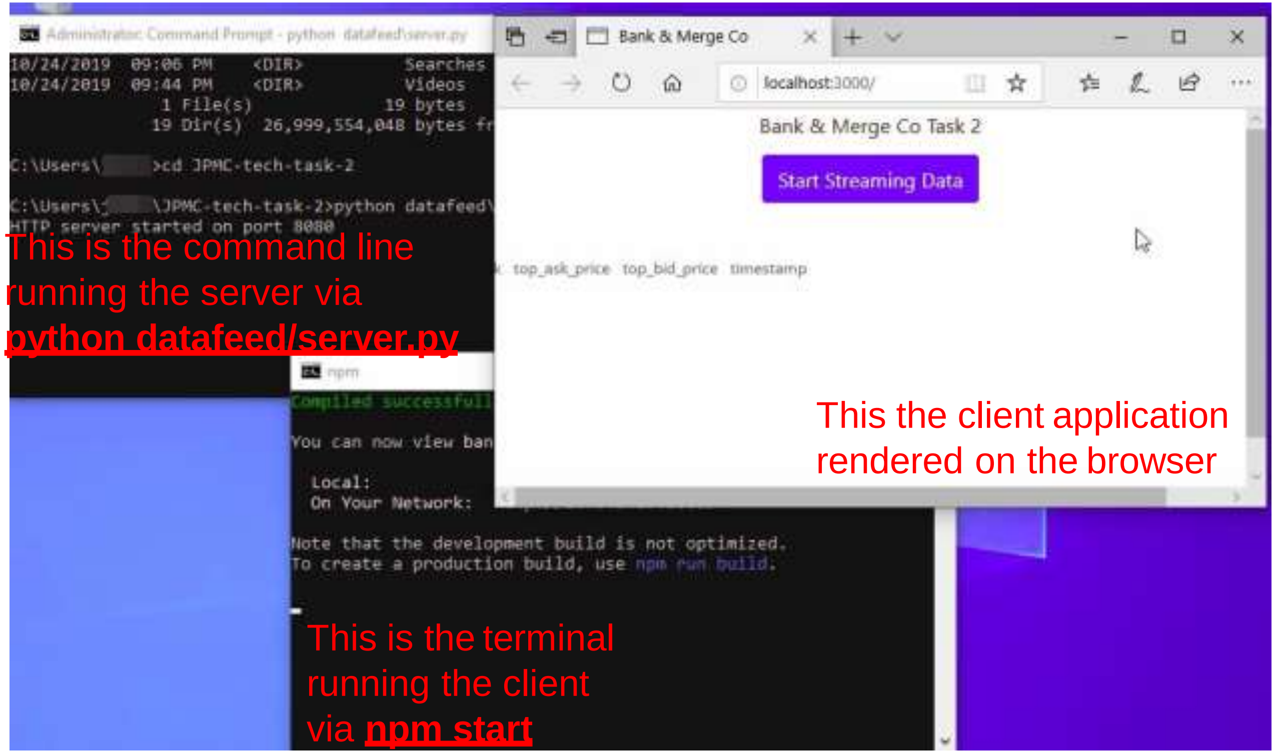
- This guide is only for those who did the setup locally on their machines.

Prerequisite

- Set up should have been done. This means, your server and client applications should have been running with no problems without introducing any changes to the code yet. You can verify this if you get a similar result to any of the following slides that include a picture of the server and client app running together

Prerequisite

Windows OS


Administrator: Command Prompt - python datafeed\server.py
Bank & Merge Co
localhost:3000/
Bank & Merge Co Task 2
Start Streaming Data
top_ask_price top_bid_price timestamp
npm

```
10/24/2019  09:06 PM    <DIR>          Searches
10/24/2019  09:44 PM    <DIR>          Videos
               1 File(s)             19 bytes
              19 Dir(s)  26,999,554,048 bytes fr

C:\Users\[redacted]>cd JPMC-tech-task-2

C:\Users\[redacted]\JPMC-tech-task-2>python datafeed\
HTTP server started on port 8080
```

```
Compiled successfull

You can now view ban

  Local:
  On Your Network:

Note that the development build is not optimized.
To create a production build, use npm run build.
```

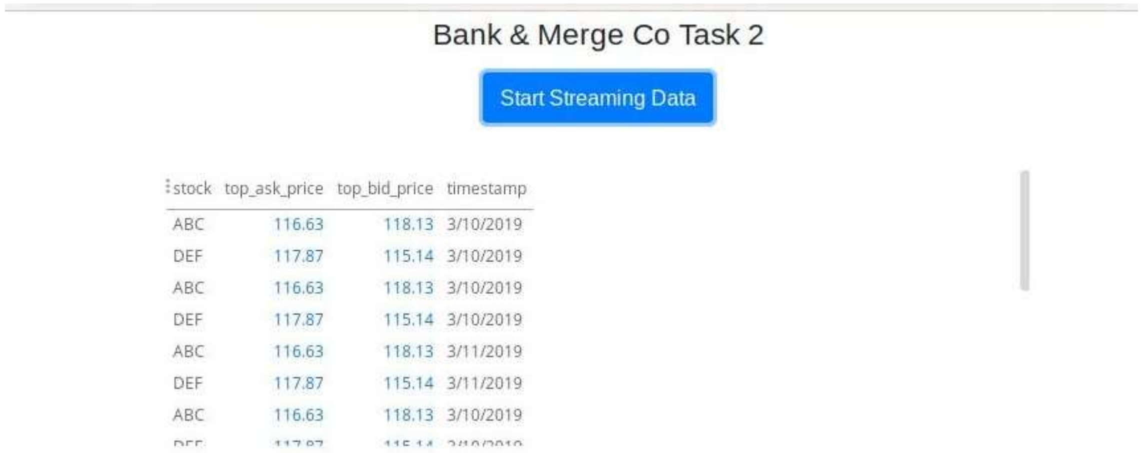
This is the command line running the server via **python datafeed/server.py**

This the client application rendered on the browser

This is the terminal running the client via **npm start**

Note: The state is similar for those who used the python3 version of the repository

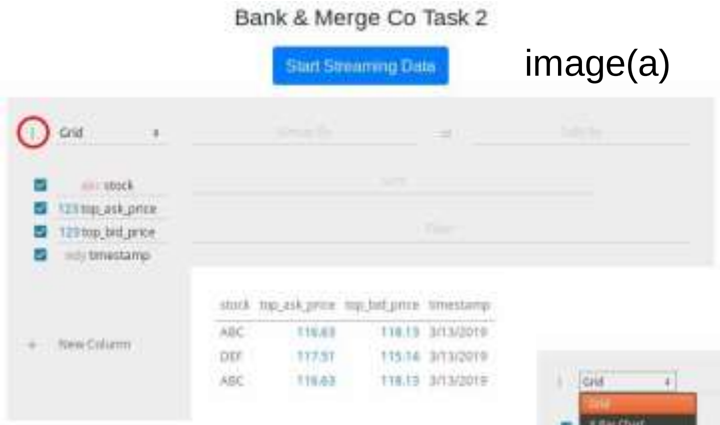
Observe Initial State Of Client App in Browser



stock	top_ask_price	top_bid_price	timestamp
ABC	116.63	118.13	3/10/2019
DEF	117.87	115.14	3/10/2019
ABC	116.63	118.13	3/10/2019
DEF	117.87	115.14	3/10/2019
ABC	116.63	118.13	3/11/2019
DEF	117.87	115.14	3/11/2019
ABC	116.63	118.13	3/10/2019

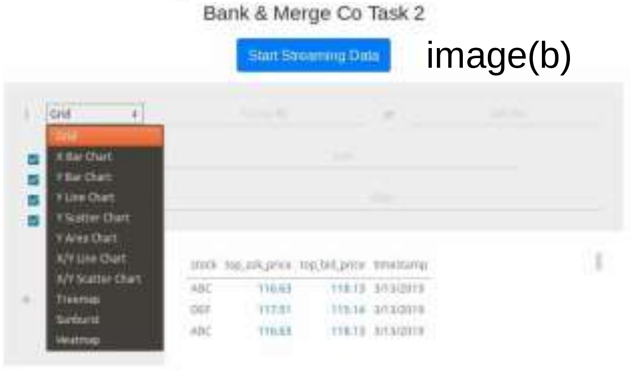
This is how the initial state of the client app looks like when you click the blue “**Start Streaming Data”** button a number of times

Observe Initial State Of Client App in Browser



stock	top_ask_price	top_bid_price	timestamp
ABC	116.63	118.13	3/13/2019
DEF	117.51	115.14	3/13/2019
ABC	116.63	118.13	3/13/2019

image(a)



stock	top_ask_price	top_bid_price	timestamp
ABC	116.63	118.13	3/13/2019
DEF	117.51	115.14	3/13/2019
ABC	116.63	118.13	3/13/2019

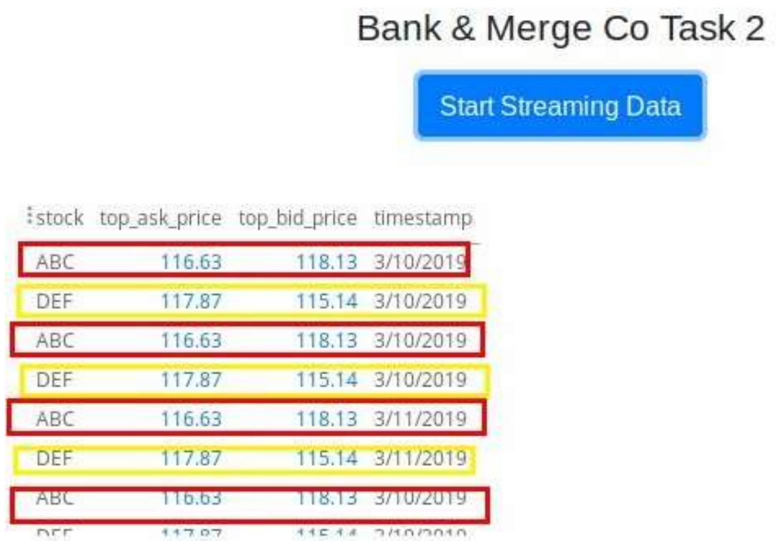
image(b)

If you clicked on the 3-dotted button on the upper left corner of the graph you'll see something like image(a) on this slide.

This tells you that the graph is configurable

Image(b) further shows you the different types of views you can use to visualize the data you have so far

Observe Initial State Of Client App in Browser



stock	top_ask_price	top_bid_price	timestamp
ABC	116.63	118.13	3/10/2019
DEF	117.87	115.14	3/10/2019
ABC	116.63	118.13	3/10/2019
DEF	117.87	115.14	3/10/2019
ABC	116.63	118.13	3/11/2019
DEF	117.87	115.14	3/11/2019
ABC	116.63	118.13	3/10/2019
DEF	117.87	115.14	3/10/2019

If looked back at the data again, you'll also observe it's just a bunch of duplicate data being printed for stocks ABC and DEF until such point that there becomes newer data i.e. different timestamp,

ask_price and **bid_price** for ABC and DEF stocks

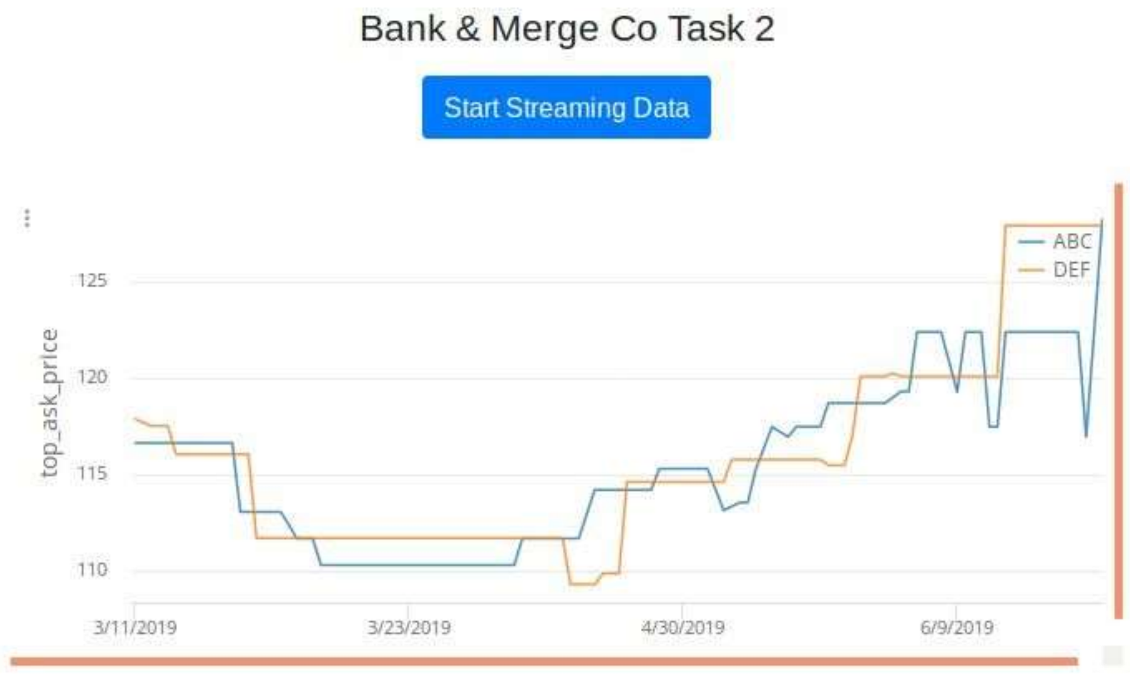
But the printing of duplicated data doesn't seem useful at all...

Objectives

- There are two things we have to achieve here to complete this task
    - (1) Make the graph continuously update instead of having to click it a bunch of times. Also the kind of graph we want to serve as visual here is kind of a continuously updating line graph whose y axis is the stock's **top_ask_price** and the x-axis is the timestamp of the stock
    - (2) Remove / disregard the duplicated data we saw earlier…

Objectives

- The kind of graph we want to up with is something like this:



Objectives

- To achieve this we have to change (2) files: **src/App.tsx** and **src/Graph.tsx**
- Don’t worry we’ll walk you through how to get these things done
- You can use any text editor your machine has and just open the files in the repository that must be changed (the guide will instruct you in the following slides which files these will be)
- Our recommendation of editors you can use would be VSCode or SublimeText as these are the most commonly used code editors out there.

Making changes in `**App.tsx**`

- App.tsx is the main app (*typescript*) file of our client side react application.

- Don’t be intimidated by words like React, which is just a javascript library to help us build interfaces and ui components, or Typescript which is just a superset of javascript but is still alike with javascript but with stronger type checking... We’ll walk you through the changes needed.

Making changes in `**App.tsx**`

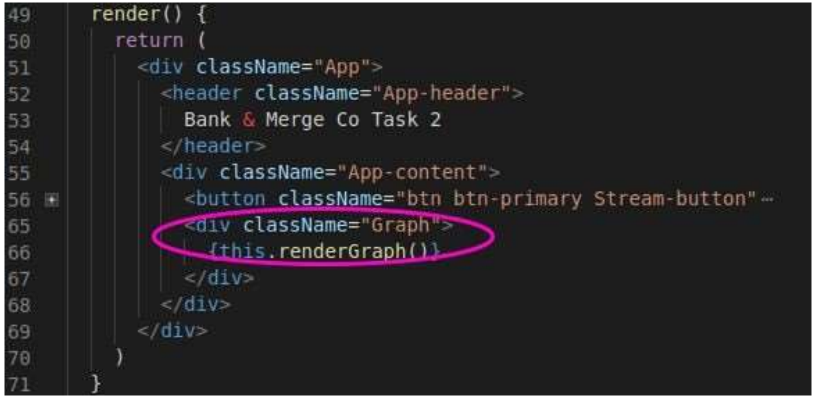
- App.tsx or the App component, is basically the first component our browser will render as it's the parent component of the other parts of the simple page that shows up when you first started the application in the set up phase.

- Components are basically the building blocks / parts of our web application. A component has a common set of properties / functions and as such, each unique component just inherits from the base React component

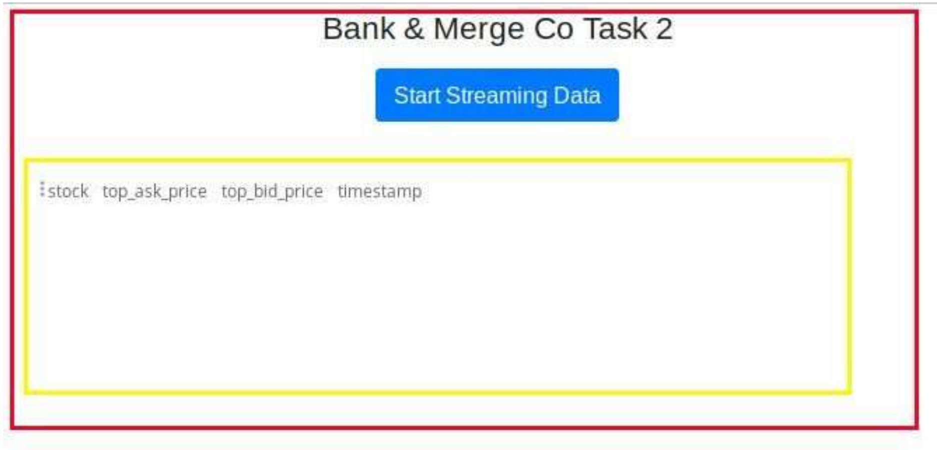
- App.tsx is the first file we will have to change to achieve our objectives

Making changes in `App.tsx`

- Red box: The App component (App.tsx)



```
render() {
  return (
    <div className="App">
      <header className="App-header">
        Bank & Merge Co Task 2
      </header>
      <div className="App-content">
        <button className="btn btn-primary Stream-button"⋯
        <div className="Graph">
          {this.renderGraph()}
        </div>
      </div>
    </div>
  )
}
```



Bank & Merge Co Task 2

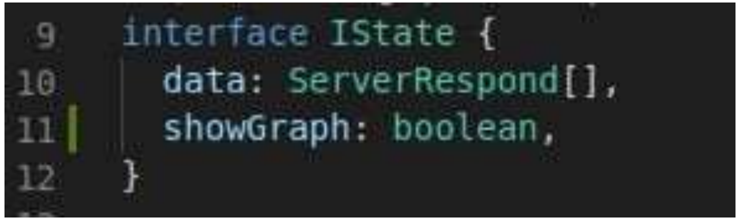
- Yellow box: The Graph component (Graph.tsx)

Making changes in `**App.tsx**`

- To achieve the objectives listed earlier, we’ll first need to make changes in App.tsx file to help us change the static table into a live / updating graph. Follow instructions in the next few slides

Making changes in `**App.tsx**`

- First you'll need to add the `showGraph` property in the **IState** interface defined in App.tsx. It should be of the type` boolean`

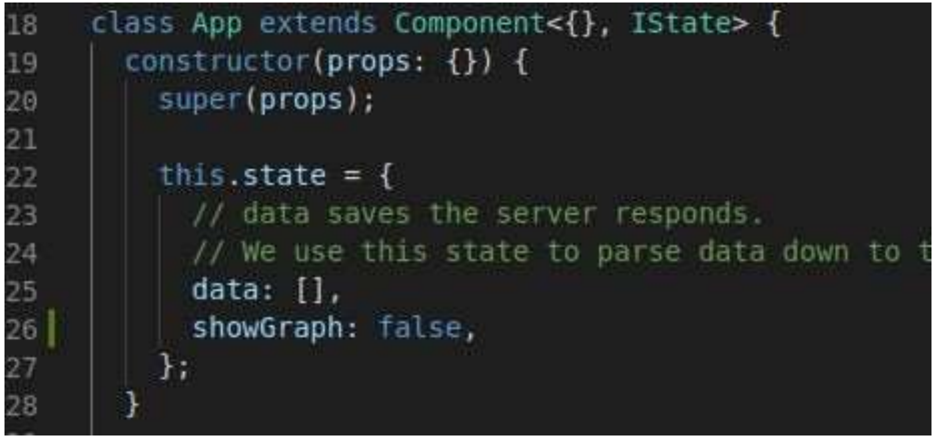


```
interface IState {
  data: ServerRespond[],
  showGraph: boolean,
}
```

*<u>note</u>: Interfaces help define the values a certain entity must have. In this case, whenever a type of **IState** is used our application knows it should always have **data** and **showGraph** as properties to be valid. If you want to learn more about interfaces in Typescript you can read [this material](#)in your spare time*

Making changes in `**App.tsx**`

- Next, in the constructor of the App component, you should define that the initial state of the App not to show the graph yet. This is because we want the graph to show when the user clicks ‘Start Streaming Data’. That means you should set `showGraph` property of the App’s state to `false` in the constructor



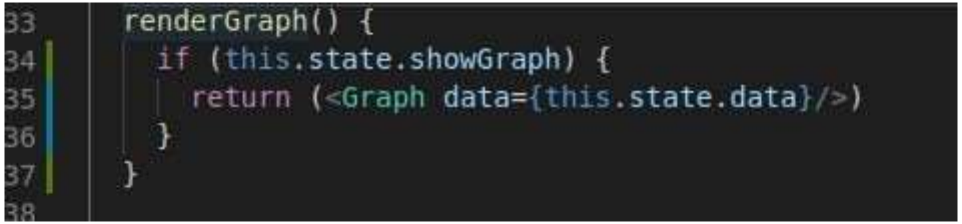
```
class App extends Component<{}, IState> {
  constructor(props: {}) {
    super(props);

    this.state = {
      // data saves the server responds.
      // We use this state to parse data down to t
      data: [],
      showGraph: false,
    };
  }
```

<u>note</u>: It’s okay if you don’t fully grasp what a constructor is yet. You can read up more on it <u>here</u>. For now, just understand it’s a method / function that’s automatically called when you create an instance of a class.

*Also, the keyword **extends** comes from <u>Object Oriented Programming Paradigm’s inheritance</u> that basically allows classes to inherit properties of another class (in this case the <u>base React Component</u>).*

Making changes in `**App.tsx**`

- To ensure that the graph doesn’t render until a user clicks the ‘Start Streaming’ button, you should also edit the `**renderGraph**` method of the App. In there, you must add a condition to only render the graph when the state’s `**showGraph**` property of the App’s state is `**true**`.



```
renderGraph() {
  if (this.state.showGraph) {
    return (<Graph data={this.state.data}/>)
  }
}
```

*note: we had to do this because **renderGraph** gets called in the **render** method of the App component. To learn more about how react renders components you can read more here*

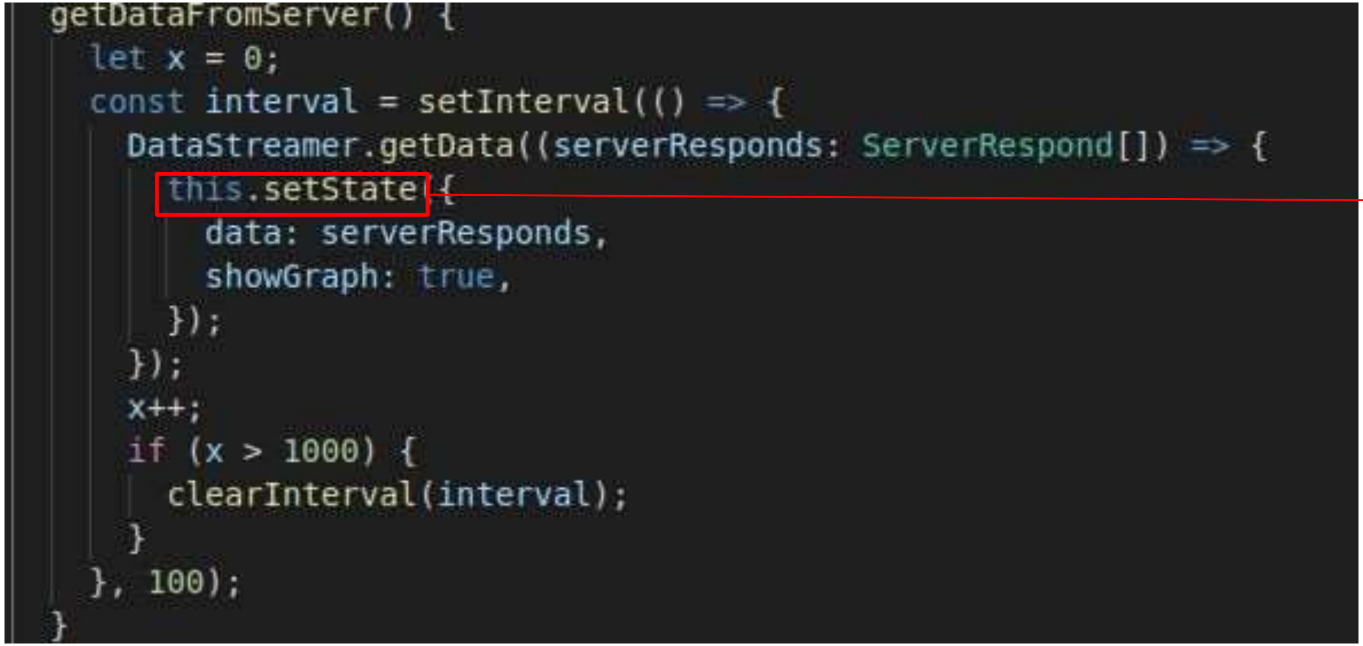
Making changes in `**App.tsx**`

- Finally, you must also modify the `**getDataFromServer**`method to contact the server and get data from it continuously instead of just getting data from it once every time you click the button.

- Javascript has a way to do things in intervals and that is via the **setInterval** function. What we can do to make it continuous (at least up to an extended period of time) is to have a guard value that we can check against when to stop / clear the interval process we started.

- You should arrive with a similar result as the next slide when you apply the modifications properly...

Making changes in `**App.tsx**`



```
getDataFromServer() {
  let x = 0;
  const interval = setInterval(() => {
    DataStreamer.getData((serverResponds: ServerRespond[]) => {
      this.setState({
        data: serverResponds,
        showGraph: true,
      });
    });
    x++;
    if (x > 1000) {
      clearInterval(interval);
    }
  }, 100);
}
```

note: the only place that you should assign the local state directly is in the constructor. Any place else in our component, you should rely on setState(). This hinges on the concept of immutability

Making changes in `**App.tsx**`

- If you noticed in the image in previous slide, it's in the same method, **getDataFromServer**, that we set **showGraph** to **true** as soon as the data from the server comes back to the requestor.

- The line **DataStreamer.getData(... => ...)** is an asynchronous process that gets the data from the server and when that process is complete, it then performs what comes after the **=>** as a callback function.

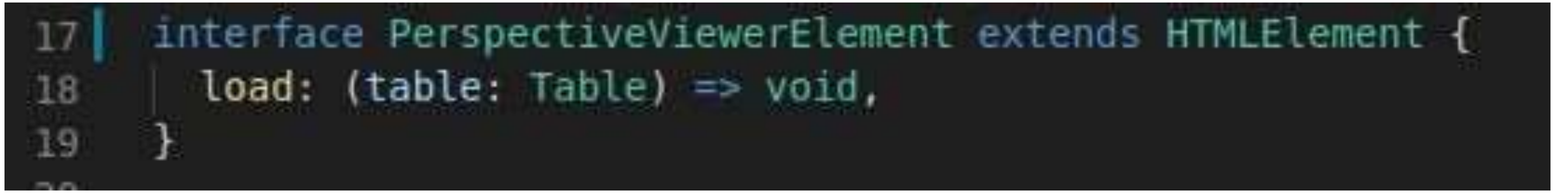
Making changes in `**App.tsx**`

- Changes in **App.tsx end** here.
- By now you should’ve accomplished modifying the client to request data from server continuously
- By now you should’ve also accomplished setting the initial state of the graph not to show until the user clicks the “**Start Streaming Data”** button
- Proceed to the next set of slides to finish the exercise with changes in **Graph.tsx**

Making changes in `**Graph.tsx**`

- To completely achieve the desired output, we must also make changes to the `**Graph.tsx**` file. This is the file that takes care of how the Graph component of our App will be rendered and react to the state changes that occur within the App.

- First, you must enable the `**PerspectiveViewerElement**` to behave like an **HTMLElement.** To do this, you can extend the `HTMLElement` class from the `**PerspectiveViewerElement**` interface.

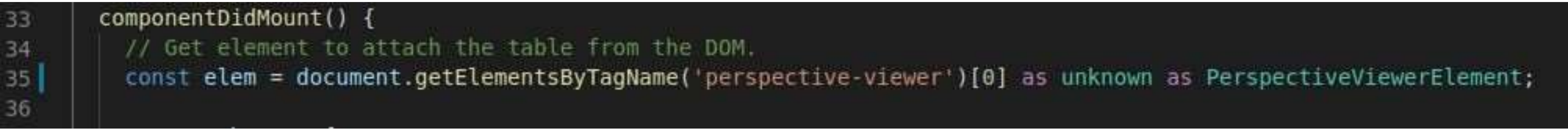


```
interface PerspectiveViewerElement extends HTMLElement {
  load: (table: Table) => void,
}
```

Making changes in `**Graph.tsx**`

- After doing this, we now need to modify `**componentDidMount**` method. Just as a note, the **componentDidMount()** method runs after the component output has been rendered to the DOM. If you want to learn more about it and other lifecycle methods/parts of react components, read more here.

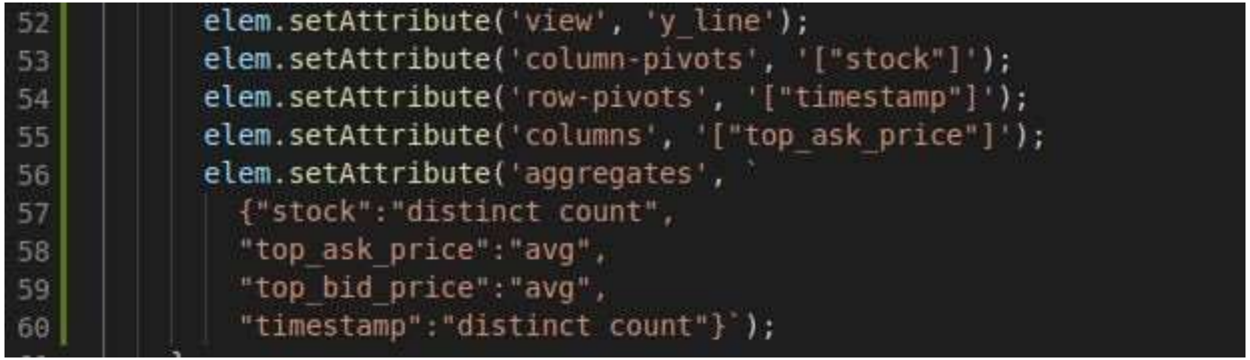
- Since you've changed the `**PerspectiveViewerElement**` to extend the `**HTMLElement**` earlier, you can now make the definition of the `**const elem**` simpler, i.e. you can just assign it straight to the result of the `**document.getElementsByTagName**`.



```
componentDidMount() {
  // Get element to attach the table from the DOM.
  const elem = document.getElementsByTagName('perspective-viewer')[0] as unknown as PerspectiveViewerElement;

```

Making changes in `Graph.tsx`

- Finally, you need to add more attributes to the element. For this you have to have read thru the Perspective configurations particularly on the table.view configurations. You'll need to add the following attributes: `**view**`, `**column-pivots**`, `**row-pivots**`, `**columns**` and `**aggregates**` . If you remember the earlier observations we did in the earlier slides, this is the configurable part of the table/graph. The end result should look something like:



```
            elem.setAttribute('view', 'y_line');
            elem.setAttribute('column-pivots', '["stock"]');
            elem.setAttribute('row-pivots', '["timestamp"]');
            elem.setAttribute('columns', '["top_ask_price"]');
            elem.setAttribute('aggregates', `
              {"stock":"distinct count",
              "top_ask_price":"avg",
              "top_bid_price":"avg",
              "timestamp":"distinct count"}`);
```

Making changes in `**Graph.tsx**`

- ‘**view**’ is the the kind of graph we wanted to visualize the data as. Initially, if you remember this was the **grid** type. However, since we wanted a continuous line graph to be the final outcome, the closest one would be **y_line**

- **‘column-pivots’** is what will allow us to distinguish stock ABC with DEF. Hence we use ‘[**“stock”**]’ as its corresponding value here. By the way, we can use stock here because it’s also defined in the **schema** object. This accessibility goes for the rest of the other attributes we’ll discuss.

- ‘**row-pivots’** takes care of our x-axis. This allows us to map each datapoint based on the timestamp it has. Without this, the x-axis is blank.

Making changes in `**Graph.tsx**`

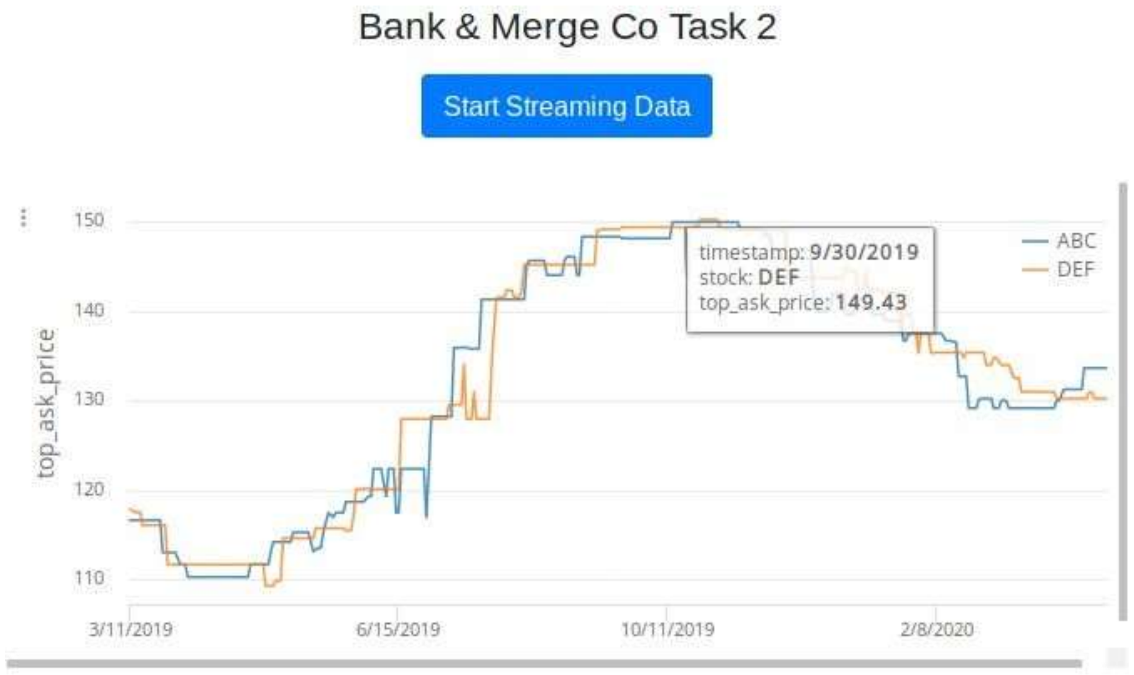
- ‘**columns’** is what will allow us to only focus on a particular part of a stock’s data along the y-axis. Without this, the graph will plot different datapoints of a stock ie: top_ask_price, top_bid_price, stock, timestamp. For this instance we only care about **top_ask_price**

- **‘aggregates’** is what will allow us to handle the duplicated data we observed earlier and consolidate them as just one data point. In our case we only want to consider a data point unique if it has a unique stock name and timestamp. Otherwise, if there are duplicates like what we had before, we will average out the top_bid_prices and the top_ask_prices of these ‘similar’ datapoints before treating them as one.

Wrapping up

- Changes in **Graph.tsx are done too**.
- By now you should've accomplished all the objectives of the task (as specified in the earlier Objectives slide) and ended up with a graph like the one in the next slide
- Feel free to poke around before completely saving everything and creating your patch file, e.g. see the different effects of changing the configurations would do to your table/graph.
- Please don't forget to leave comments in your code especially at the places where the fixes for bugs and where you piped in the data feed - this will help with other team member's understanding of your work.

End Result


Bank & Merge Co Task 2

InsideSherpa

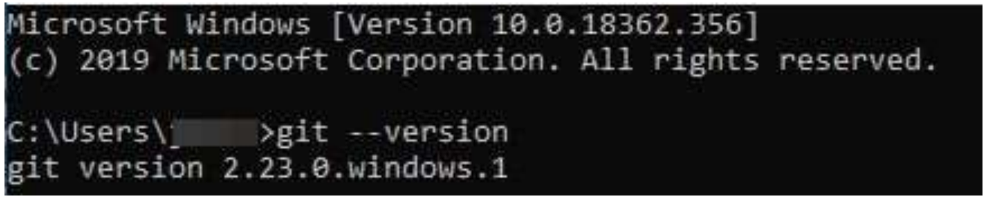
JPMorgan Chase Software Engineering Virtual Experience

Setting up your Windows for the JPMorgan Chase program

Module 3 - Display data visually for traders

Local Setup (Windows)

- First you must have git installed in your system. Git is usually used by programmers to collaborate with code in a software project.To do this, follow this quick guide. You know you have installed successfully when you get this output on your command line (cmd). (*any git version should suffice but the latest is recommended)*



```
Microsoft Windows [Version 10.0.18362.356]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\[redacted]>git --version
git version 2.23.0.windows.1
```

Local Setup (Windows)

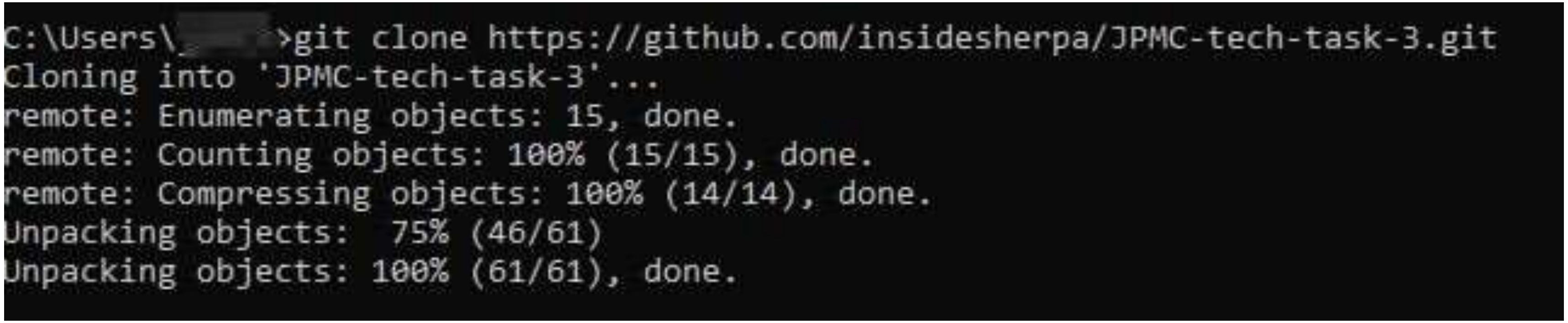
- Once you have git installed, you need a copy of the application code you'll be working with on your machine. To do this, you must execute the following commands on your terminal:

```
git clone https://github.com/insidesherpa/JPMC-tech-task-3.git
git clone https://github.com/insidesherpa/JPMC-tech-task-3-PY3.git
```

- This command will download the code repositories from github to your machine in the current working directory of the terminal you executed the command in. Downloading the 2 repositories above will give you options later

Local Setup (Windows)

- You’ll know you cloned successfully if you have the copy of the application code on your machine:



```
C:\Users\[redacted]>git clone https://github.com/insidesherpa/JPMC-tech-task-3.git
Cloning into 'JPMC-tech-task-3'...
remote: Enumerating objects: 15, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (14/14), done.
Unpacking objects:  75% (46/61)
Unpacking objects: 100% (61/61), done.
```

note: the image here just does not contain the other repository but it should if you did the previous slides and execute the ***dir*** *command. `dir` just lists the files/folders in the current directory*

Local Setup (Windows)

- To access the files inside from the terminal, just change directory by typing:

```
cd JPMC-tech-task-3
```

note: If you choose to work using python3 and your system has version python3 or above instead of python2.7.x, then choose to go into the other repository you downloaded instead. (otherwise, use the other repo above); ***cd*** *changes directory your terminal is in. For more info on how to use cd, check here*

```
cd JPMC-tech-task-3-py3
```

Local Setup (Windows)

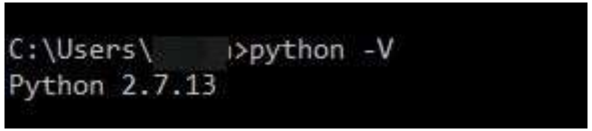
- To clarify, you're only supposed to work on one of the repositories you cloned / downloaded into your system. It all depends on what Python version you primarily use

- Python is just a scripting / programming language we developers use quite often in the field. This application you'll be working on uses it.

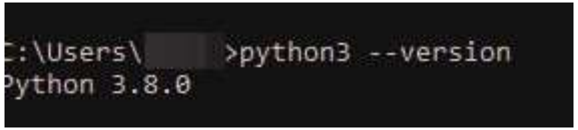
- We'll discuss checking / installing Python in your system in the following slides

Local Setup (Windows)

- Next, you’ll need to have Python 2.7 **and/or** Python 3+ installed on your machine. Follow the instructions here (for python2), **and/or** here (for python3) You can verify this on your command line (cmd) if you get a result like:



```
C:\Users\[redacted]>python -V
Python 2.7.13
```



```
:\Users\[redacted]>python3 --version
ython 3.8.0
```

(any python 2.7.x > = 2.7.16 should suffice
but the latest 2.7.x is recommended (**2.7.17**)*;*
any python 3.x >= 3.6is fine, latest is recommended (**3.8.0**))

Execute the command below to verify what version you have:

```
python --version
```

Note: the image here is only of 2.7 but it should be similar if you check for python3. If you install both versions, make sure the command **python** maps to 2.7.x at least

Sometimes your system might have it as

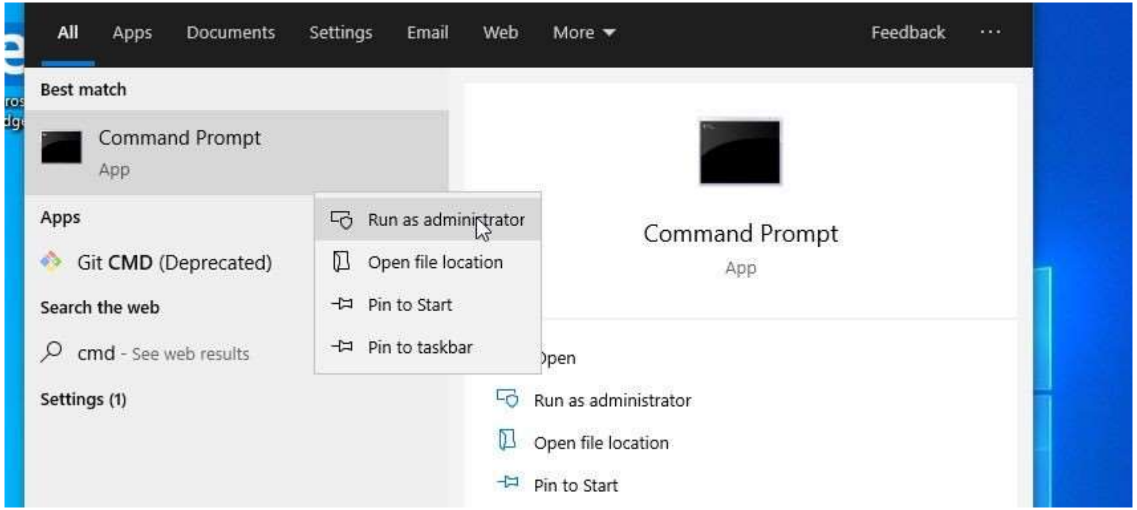
```
python3 --version
```

Local Setup (Windows)

- If something went wrong with the output you got from the previous slide, please check the Troubleshooting slides for Windows in this guide.

Local Setup (Windows)

- Once you have python is installed, all you have to do get the server up and running is to start the server in its own cmd. Ensure that the command line wherein you run the server app is on Administrator mode:



Local Setup (Windows)

- Once you have Python 2.7 and/or Python 3 installed, you can start the server application in one terminal by just executing it:
  (note: just choose to run one server; either the python 2 or python 3 version of server. Run the commands below depending on your python version)

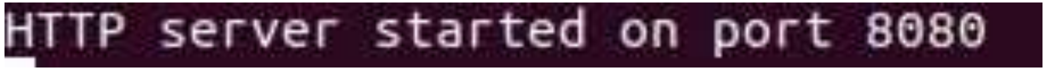
```
// If python --version = 2.7+, you must be in the JPMC-tech-task-3
// If python --version = 3+ , you must be in JPMC-tech-task-3-py3 directory
python datafeed/server.py
```

```
// If your system makes the distinction of python3 as `python3`,
// you must be in JPMC-tech-task-3-py3 directory
python3 datafeed/server3.py
```

If ever you encounter an error when starting the server application, see troubleshooting in this slide

Local Setup (Windows)

- If you've done the previous slide, then you should get something similar to the pic below when you ran the server.



```
HTTP server started on port 8080
```

- To be clear, the server application isn't stuck. It's behaving perfectly normal here. The reason why it's just like that for now is because it's just listening for requests

- For us to be able to make requests, we have to start the client application. The following slides will help you do that.

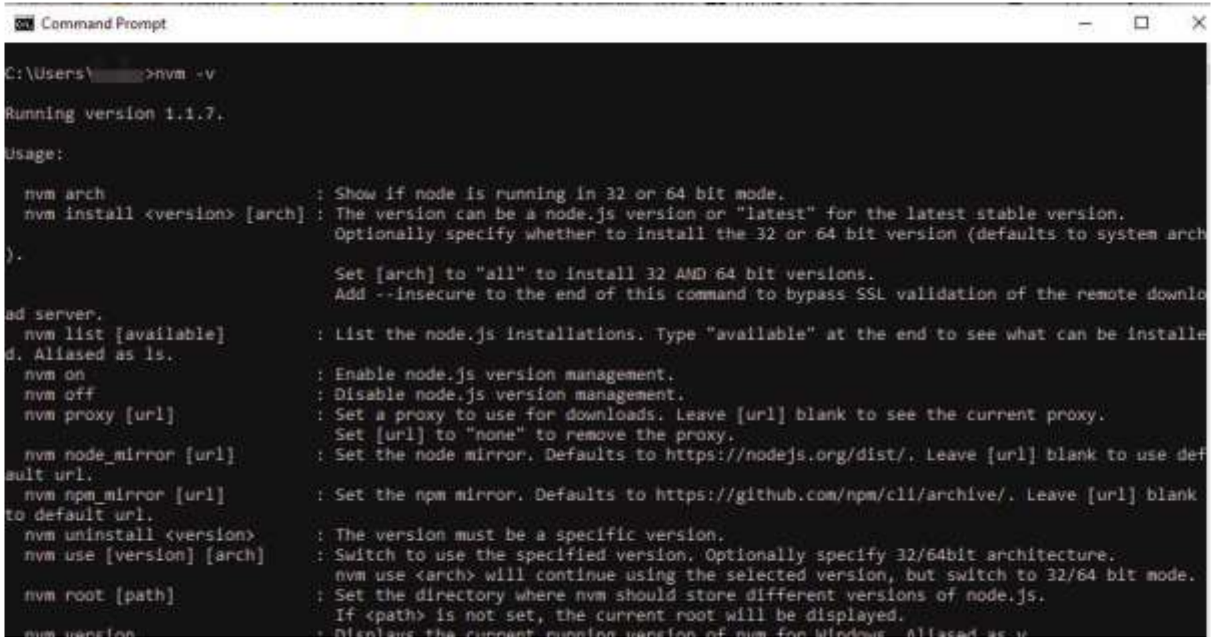
Local Setup (Windows)

- In a separate command line, let’s install the other remaining dependencies i.e. Node and Npm. Node is a JavaScript runtime environment that executes JavaScript code outside of a browser and Npm is node’s package manager that helps us to install other libraries/dependencies we’ll be using in our web application app.

- To install node and npm and be able to manage versions seamlessly we will install NVM (node version manager). Once this is on your machine you can basically install any version of node and npm and switch depending on a project’s needs. Follow these instructions to install nvm for windows.

Local Setup (Windows)

- You will know you've successfully installed nvm if you get a similar result below when you type the command **nvm -v**



Command Prompt

```
C:\Users\[redacted]>nvm -v

Running version 1.1.7.

Usage:

  nvm arch                     : Show if node is running in 32 or 64 bit mode.
  nvm install <version> [arch] : The version can be a node.js version or "latest" for the latest stable version.
                                 Optionally specify whether to install the 32 or 64 bit version (defaults to system arch
).
                                 Set [arch] to "all" to install 32 AND 64 bit versions.
                                 Add --insecure to the end of this command to bypass SSL validation of the remote downlo
ad server.
  nvm list [available]         : List the node.js installations. Type "available" at the end to see what can be installe
d. Aliased as ls.
  nvm on                       : Enable node.js version management.
  nvm off                      : Disable node.js version management.
  nvm proxy [url]              : Set a proxy to use for downloads. Leave [url] blank to see the current proxy.
                                 Set [url] to "none" to remove the proxy.
  nvm node_mirror [url]        : Set the node mirror. Defaults to https://nodejs.org/dist/. Leave [url] blank to use def
ault url.
  nvm npm_mirror [url]         : Set the npm mirror. Defaults to https://github.com/npm/cli/archive/. Leave [url] blank
to default url.
  nvm uninstall <version>      : The version must be a specific version.
  nvm use [version] [arch]     : Switch to use the specified version. Optionally specify 32/64bit architecture.
                                 nvm use <arch> will continue using the selected version, but switch to 32/64 bit mode.
  nvm root [path]              : Set the directory where nvm should store different versions of node.js.
                                 If <path> is not set, the current root will be displayed.
  nvm version                  : Displays the current running version of nvm for Windows. Aliased as v.
```

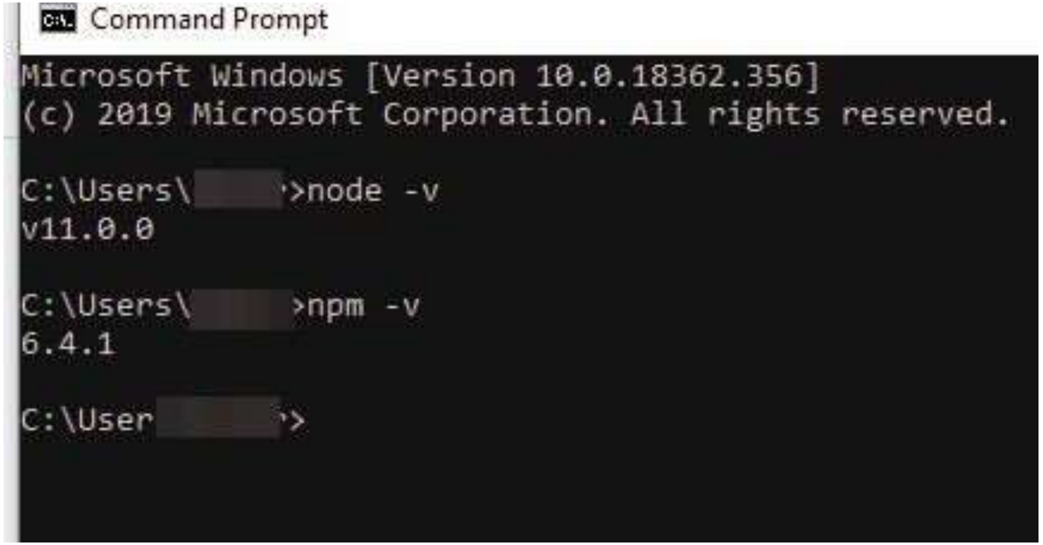
Local Setup (Windows)

- Now, we just need to install the right node version using nvm and that would consequently get us the right npm version as well. We do this by executing the commands:

```
nvm install v11.0.0

nvm use v11.0.0
```

- You should end up with a similar result in the next slide after doing the commands above

Local Setup (Windows)


Command Prompt

```
Microsoft Windows [Version 10.0.18362.356]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\[redacted]>node -v
v11.0.0

C:\Users\[redacted]>npm -v
6.4.1

C:\User[redacted]>
```

To check your node version type and enter:

node -v

To check your npm version type and enter:

npm -v

<u>Take note:</u> If you open a new terminal after all this, make sure to recheck your node version and npm version. It might be the case you've switched to a different version so just execute `nvm use v11.0.0` again if ever...

Local Setup (Windows)

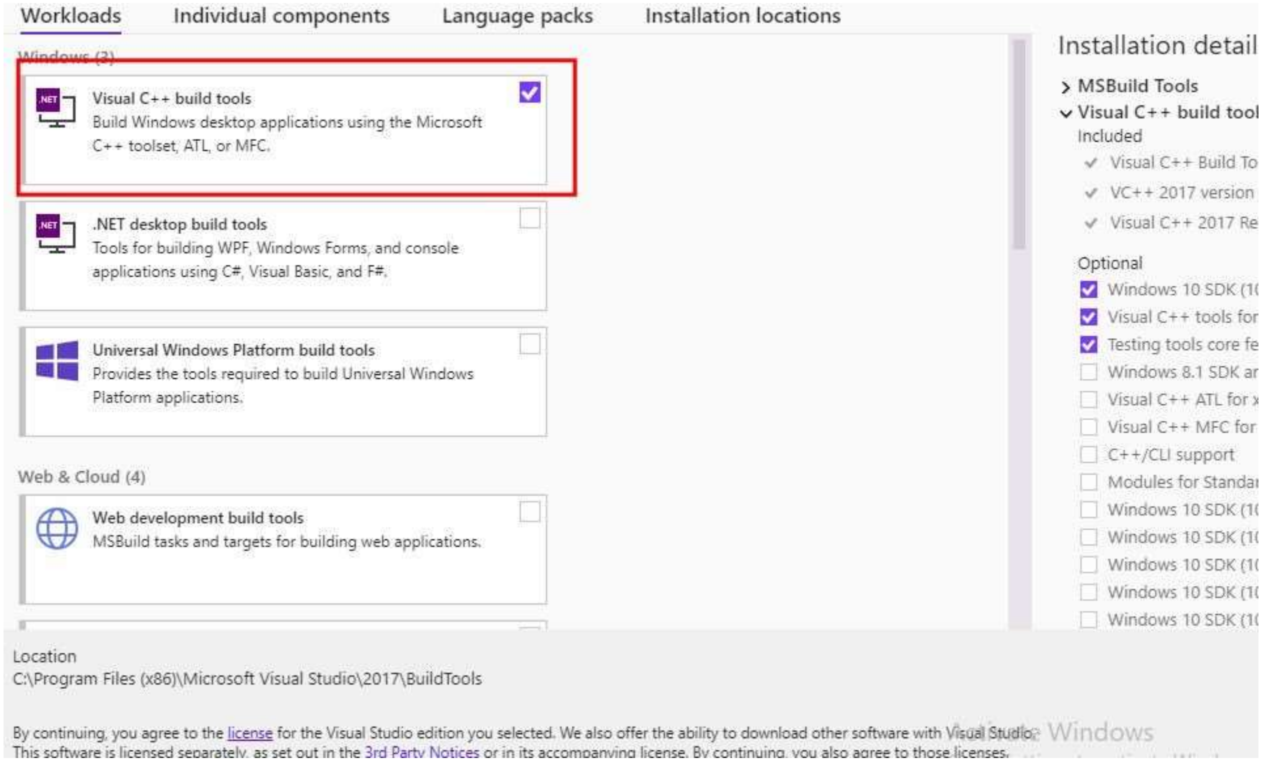
- Because windows' nvm isn't exactly like mac's or linux's, there's still a couple more dependencies we have to install in order to get this whole application to work.

- We have to install Visual C++ Build Environment via Visual Studio Build Tools. Run the downloaded .exe file and make sure to have the basics installed i.e "Visual C++ Build Tools on your machine. This is actually need because of node-gyp. After installing, make sure to run in your command line:

  ```
  npm config set msvs_version 2017
  ```

  (*see image in the next slide*)

Local Setup (Windows)



Windows (3)

- [✓] **Visual C++ build tools** — Build Windows desktop applications using the Microsoft C++ toolset, ATL, or MFC.
- [ ] **.NET desktop build tools** — Tools for building WPF, Windows Forms, and console applications using C#, Visual Basic, and F#.
- [ ] **Universal Windows Platform build tools** — Provides the tools required to build Universal Windows Platform applications.

Web & Cloud (4)

- [ ] **Web development build tools** — MSBuild tasks and targets for building web applications.

Installation detail

> MSBuild Tools

∨ Visual C++ build tool

Included

- ✓ Visual C++ Build To
- ✓ VC++ 2017 version
- ✓ Visual C++ 2017 Re

Optional

- [✓] Windows 10 SDK (1
- [✓] Visual C++ tools for
- [✓] Testing tools core fe
- [ ] Windows 8.1 SDK ar
- [ ] Visual C++ ATL for x
- [ ] Visual C++ MFC for
- [ ] C++/CLI support
- [ ] Modules for Standar
- [ ] Windows 10 SDK (1
- [ ] Windows 10 SDK (1
- [ ] Windows 10 SDK (1
- [ ] Windows 10 SDK (1
- [ ] Windows 10 SDK (1

Location
C:\Program Files (x86)\Microsoft Visual Studio\2017\BuildTools

By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

This is how installing visual C++ build environment would look like when you run the .exe file

Local Setup (Windows)

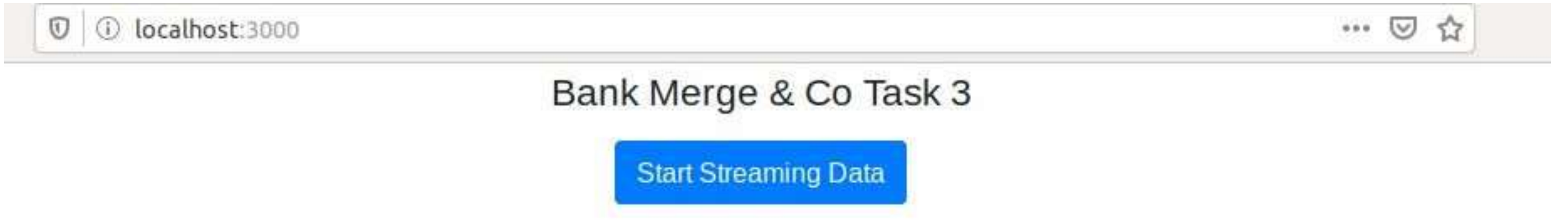
- Finally, to start the client application, all we have to do would be to run the commands below

```
npm install
```

  npm start

- If all goes well , you should end up with a similar result in the next slide. If you you encounter problems with npm install, particularly relating to node-gyp try the other windows options of installing here. If there are other issues, please refer to the Troubleshooting slides in this guide.

Local Setup (Windows)



Note: This part assumes no errors came out of **npm install.** Some data should show if you click on the "Start Streaming Data" button. If nothing is showing, check the command line where you're running the server and see if anything printed out, like "Query received". If there's something like that and you're not seeing results, then try using a different browser (e.g. Chrome/Firefox) and checking localhost:3000. If no response like "Query received" got printed in the command line running the server, you might be experiencing this issue.

Local Setup: Troubleshooting (Windows)

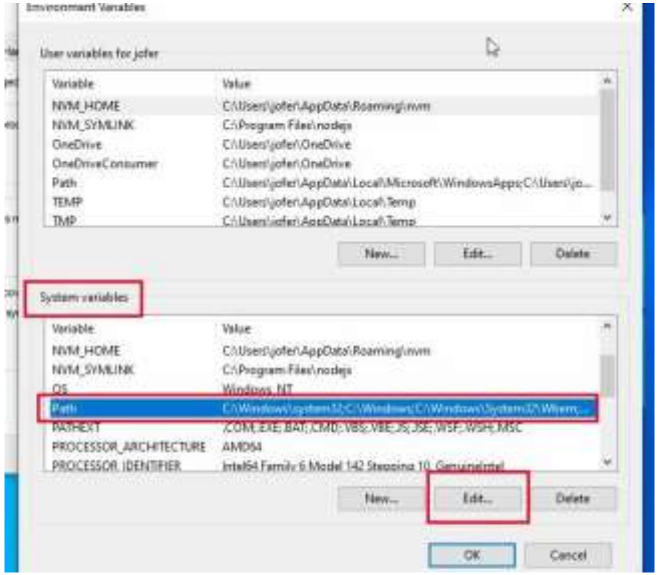
- If you did not encounter any issues, your setup is finished. From here on, you can make changes to the code and eventually arrive at the desired output.

- If you did encounter issues, check if the commonly encountered issues listed in the next few slides will solve your problem:
    - python not recognized or not returning in your command line
    - dateutil dependency
    - npm install or npm start errors
    - Socket unavailable

Local Setup: Troubleshooting (Windows)

- Click that and click “Environment Variables” in the next window (image on left side)

- You should then come up with the image below:



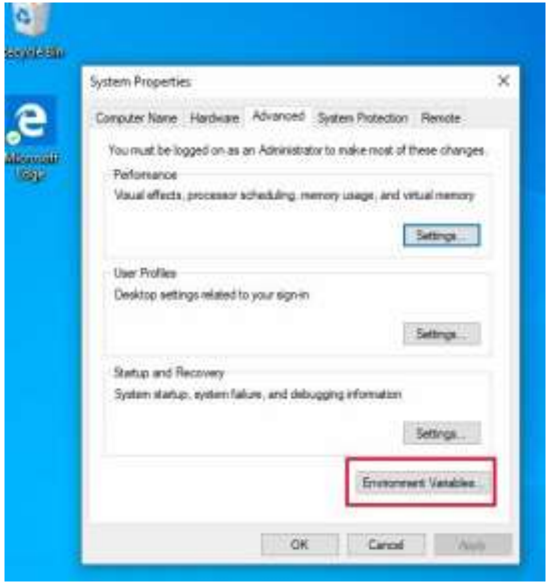
Environment Variables

User variables for jofer

Variable	Value
NVM_HOME	C:\Users\jofer\AppData\Roaming\nvm
NVM_SYMLINK	C:\Program Files\nodejs
OneDrive	C:\Users\jofer\OneDrive
OneDriveConsumer	C:\Users\jofer\OneDrive
Path	C:\Users\jofer\AppData\Local\Microsoft\WindowsApps;C:\Users\jo...
TEMP	C:\Users\jofer\AppData\Local\Temp
TMP	C:\Users\jofer\AppData\Local\Temp

System variables

Variable	Value
NVM_HOME	C:\Users\jofer\AppData\Roaming\nvm
NVM_SYMLINK	C:\Program Files\nodejs
OS	Windows_NT
Path	C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 142 Stepping 10, GenuineIntel



System Properties

Computer Name | Hardware | Advanced | System Protection | Remote

You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory

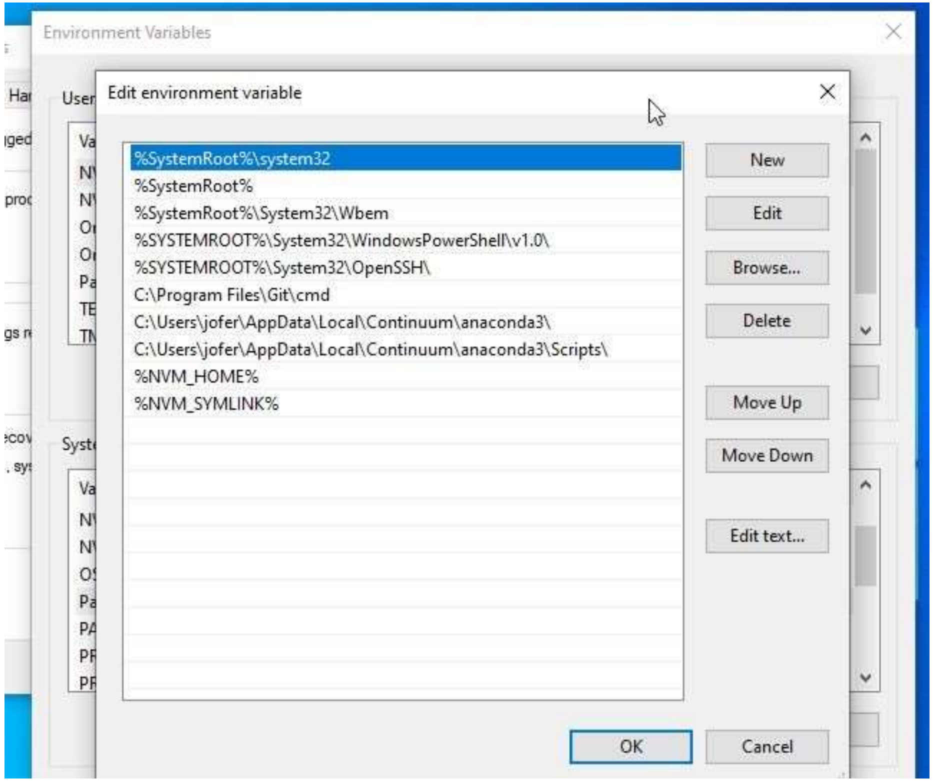
User Profiles
Desktop settings related to your sign-in

Startup and Recovery
System startup, system failure, and debugging information

You want to edit the **Path** under **System variables**

Local Setup: Troubleshooting (Windows)

- You should end up with a window like this after the steps from the previous slide:



```
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\Git\cmd
C:\Users\jofer\AppData\Local\Continuum\anaconda3\
C:\Users\jofer\AppData\Local\Continuum\anaconda3\Scripts\
%NVM_HOME%
%NVM_SYMLINK%
```

You want to make sure the directory where the **Python** executable/application you want to use is in is included in the list of paths.

If you followed how to install **Python2.7.x** in the earlier slides, you should have it in C:\\Python27**.** Make sure to include that in your path.

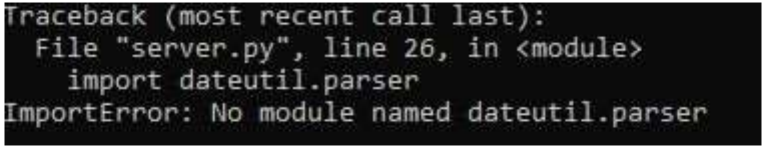
Same goes for **Python3.x** if you want to enable it. But make sure to remove your Python2.7.x path first .

If you installed python using other means, e.g. anaconda, its python executable is located elsewhere but same method of putting the path applies

Don’t forget to restart your command line after setting a new path to reflect the changes...

Local Setup: Troubleshooting (Windows)

- In some cases, dependency issues might arise like when you run `server.py`:



```
Traceback (most recent call last):
  File "server.py", line 26, in <module>
    import dateutil.parser
ImportError: No module named dateutil.parser
```

In this case, you must install pip first. pip is python's package manager for installing python dependencies. Make sure you install pip for the right Python version you're working with in this project. You can check your pip version by **pip --version** and it will tell which python version it maps too

Local Setup: Troubleshooting (Windows)

- Installing pip nowadays usually involves downloading the get-pip.py script. If you followed the instructions in the last slide, it usually involves using the command:

  ```
  curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
  ```

  If you don't have curl, just copy a python file, rename it to get-pip.py and replace all the contents of the python file to what's in get-pip.py

  ```
  //if python --version = 2.7+ this will install pip for python2
  //if python --version = 3+ this will install pip for python3
  python get-pip.py

  //if your system makes the distinction of python3 as `python3` then
  //doing the command below will install pip for python3
  python3 get-pip.py
  ```

Local Setup: Troubleshooting (Windows)

- After that, for pip to be a recognizable command in your terminal/command line, you need to add it in your system's path environment variable
  - On Windows, for Python2.7 it's usually in C:\\Python27\Scripts. It would also be similar for Python3.x if you followed the installation guide for Python earlier (e.g. C:\\Python3X\Scripts)

  - Make sure you open a new command line too and use that instead after doing this

- To edit your system path environment variable it's similar to the slides here.
- Alternatively you can access it doing something like:
  - **C:\\Python27\Scripts\pip.exe <parameters>** (similar for python3x if it was installed in C:\\)
  - <parameters> could be something like **C:\\Python27\Scripts\pip.exe install python-dateutil**
  - Take note though, this assumes that you have your python installed in drive C:\\

Local Setup: Troubleshooting (Windows)

- Then afterwards, you can run the following command on your terminal to install the dependency:

```
pip install python-dateutil
```

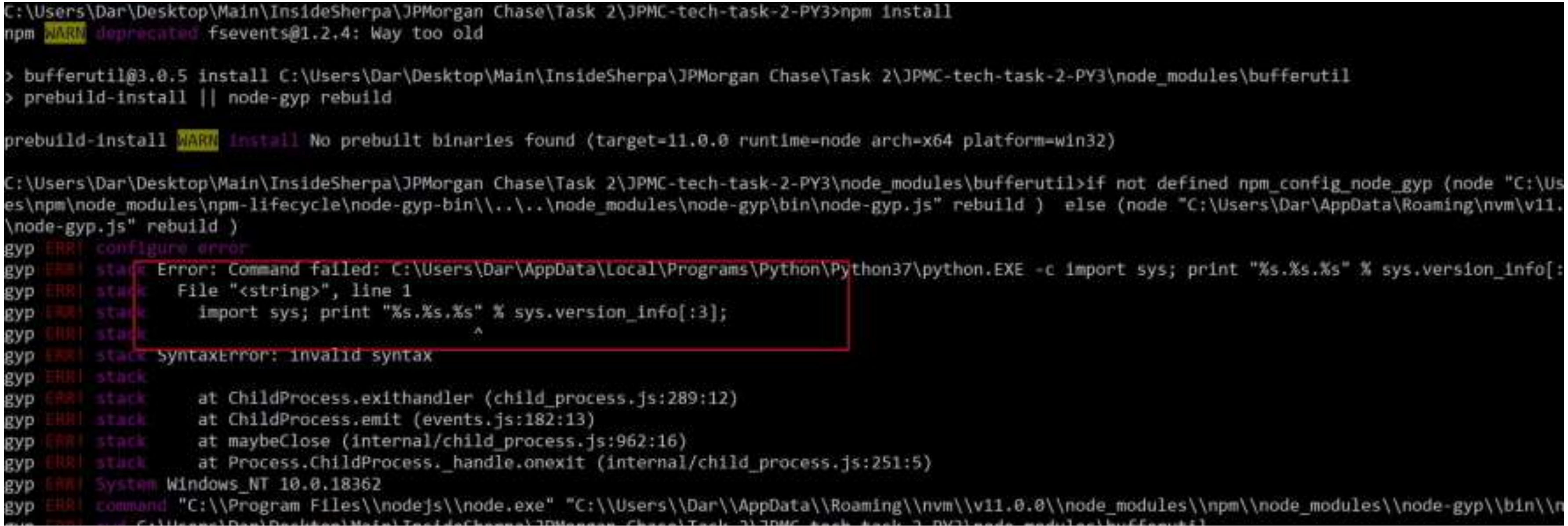
Afterwards, you can rerun the server and then rerun the client

Note: For the command above, whatever python version your pip corresponds to (i.e. the output of **pip --version**, that is the python version that will have the dependency installed). So if you're **pip** corresponds to python2.7.x then doing the command above will install python-dateutil for python2.7.x

Local Setup: Troubleshooting (Windows)

- In other cases, you might encounter problems after doing **npm install**. Errors like the following might appear on your end

CASE A:



```
C:\Users\Dar\Desktop\Main\InsideSherpa\JPMorgan Chase\Task 2\JPMC-tech-task-2-PY3>npm install
npm WARN deprecated fsevents@1.2.4: Way too old

> bufferutil@3.0.5 install C:\Users\Dar\Desktop\Main\InsideSherpa\JPMorgan Chase\Task 2\JPMC-tech-task-2-PY3\node_modules\bufferutil
> prebuild-install || node-gyp rebuild

prebuild-install WARN install No prebuilt binaries found (target=11.0.0 runtime=node arch=x64 platform=win32)

C:\Users\Dar\Desktop\Main\InsideSherpa\JPMorgan Chase\Task 2\JPMC-tech-task-2-PY3\node_modules\bufferutil>if not defined npm_config_node_gyp (node "C:\Us
es\npm\node_modules\npm-lifecycle\node-gyp-bin\\..\..\node_modules\node-gyp\bin\node-gyp.js" rebuild )  else (node "C:\Users\Dar\AppData\Roaming\nvm\v11.
\node-gyp.js" rebuild )
gyp ERR! configure error
gyp ERR! stack Error: Command failed: C:\Users\Dar\AppData\Local\Programs\Python\Python37\python.EXE -c import sys; print "%s.%s.%s" % sys.version_info[:
gyp ERR! stack   File "<string>", line 1
gyp ERR! stack     import sys; print "%s.%s.%s" % sys.version_info[:3];
gyp ERR! stack                                  ^
gyp ERR! stack SyntaxError: invalid syntax
gyp ERR! stack
gyp ERR! stack     at ChildProcess.exithandler (child_process.js:289:12)
gyp ERR! stack     at ChildProcess.emit (events.js:182:13)
gyp ERR! stack     at maybeClose (internal/child_process.js:962:16)
gyp ERR! stack     at Process.ChildProcess._handle.onexit (internal/child_process.js:251:5)
gyp ERR! System Windows_NT 10.0.18362
gyp ERR! command "C:\\Program Files\\nodejs\\node.exe" "C:\\Users\\Dar\\AppData\\Roaming\\nvm\\v11.0.0\\node_modules\\npm\\node_modules\\node-gyp\\bin\\n
```

Local Setup: Troubleshooting (Windows)

- **CASE A**: If you’re having a problem similar to this situation, most likely, you haven’t set **python** in npm to map to **python2.7.x**. Make sure you’ve installed **python 2.7.x** first and make sure you map the **python** command npm uses to **python 2.7.x** . Here’s a reference on how to do it.

- In our setup node-gyp is called by way of npm, so in order for you to avoid the error, you have to explicitly tell npm what python version to use. Thus, you have to set npm's 'python' config key to the appropriate value:
    - **npm config set python /path/to/executable/python** where /path/to/executable/python is the directory where the python2.7.x binary is, for example in /usr/bin/python usually in linux/mac or C:\\Python27 for Windows

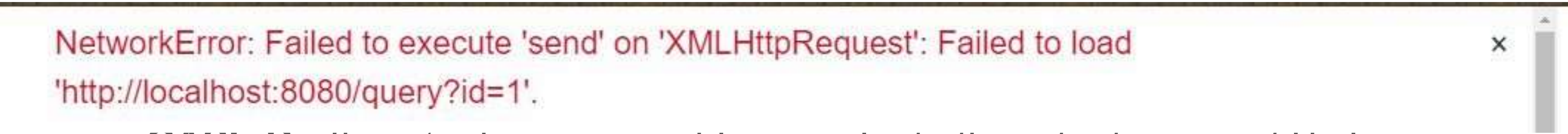
Local Setup: Troubleshooting (Windows)

- **CASE A (continuation)**: Alternatively, you can set **python** in your system to map to **python2.7.x**. Make sure you've installed **python 2.7.x** first and make sure you map the **python** command to **python 2.7.x** (meaning **python --version** should output 2.7.x) Check the slides on how to set this via the system environment path variable in the earlier slides.

- It could help that you also temporarily remove in that same path variable the mapping to **python3.x** if you also have it in your system. Just place it back after you **npm install** successfully and if you plan on using the **py3** version of the repository...

Local Setup: Troubleshooting (Windows)

CASE B:



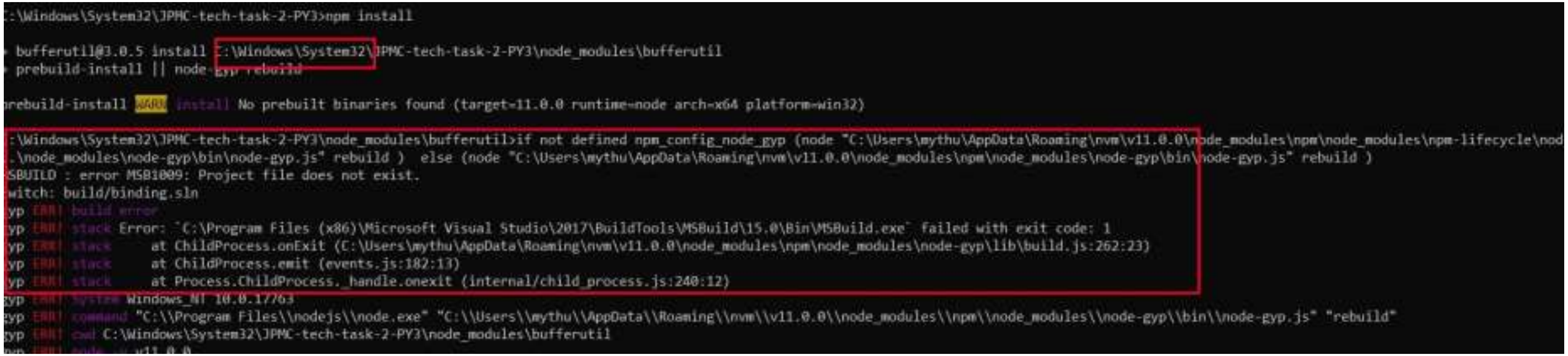
```
NetworkError: Failed to execute 'send' on 'XMLHttpRequest': Failed to load
'http://localhost:8080/query?id=1'.
```

- **CASE B:** If you´re having a problem similar to this situation, most likely, you aren’t running the server application alongside the client app. Make sure you have another terminal open wherein you’re running the server app which is run via **python datafeed/server.py** or **python datafeed/server3.py**

  If running the server app errors out for you might be experiencing this other problem (see linked slide)

Local Setup: Troubleshooting (Windows)

CASE C:



```
:\Windows\System32\JPMC-tech-task-2-PY3>npm install

 bufferutil@3.0.5 install C:\Windows\System32\JPMC-tech-task-2-PY3\node_modules\bufferutil
 prebuild-install || node-gyp rebuild

rebuild-install WARN install No prebuilt binaries found (target=11.0.0 runtime=node arch=x64 platform=win32)

:\Windows\System32\JPMC-tech-task-2-PY3\node_modules\bufferutil>if not defined npm_config_node_gyp (node "C:\Users\mythu\AppData\Roaming\nvm\v11.0.0\node_modules\npm\node_modules\npm-lifecycle\nod
.\node_modules\node-gyp\bin\node-gyp.js" rebuild )  else (node "C:\Users\mythu\AppData\Roaming\nvm\v11.0.0\node_modules\npm\node_modules\node-gyp\bin\node-gyp.js" rebuild )
SBUILD : error MSB1009: Project file does not exist.
witch: build/binding.sln
yp ERR! build error
yp ERR! stack Error: `C:\Program Files (x86)\Microsoft Visual Studio\2017\BuildTools\MSBuild\15.0\Bin\MSBuild.exe` failed with exit code: 1
yp ERR! stack     at ChildProcess.onExit (C:\Users\mythu\AppData\Roaming\nvm\v11.0.0\node_modules\npm\node_modules\node-gyp\lib\build.js:262:23)
yp ERR! stack     at ChildProcess.emit (events.js:182:13)
yp ERR! stack     at Process.ChildProcess._handle.onexit (internal/child_process.js:240:12)
gyp ERR! System Windows_NT 10.0.17763
gyp ERR! command "C:\\Program Files\\nodejs\\node.exe" "C:\\Users\\mythu\\AppData\\Roaming\\nvm\\v11.0.0\\node_modules\\npm\\node_modules\\node-gyp\\bin\\node-gyp.js" "rebuild"
gyp ERR! cwd C:\Windows\System32\JPMC-tech-task-2-PY3\node_modules\bufferutil
```

Case C: This problem is most likely due to the fact that you’re in C:\\Windows\System32. Do not clone the repo here as mentioned. Clone in C:\\Users\<some_account>\ instead.

Local Setup: Troubleshooting (Windows)

- After trying the earlier suggestions for troubleshooting and **npm install** still errors out for you, try downloading the **node_modules** here.

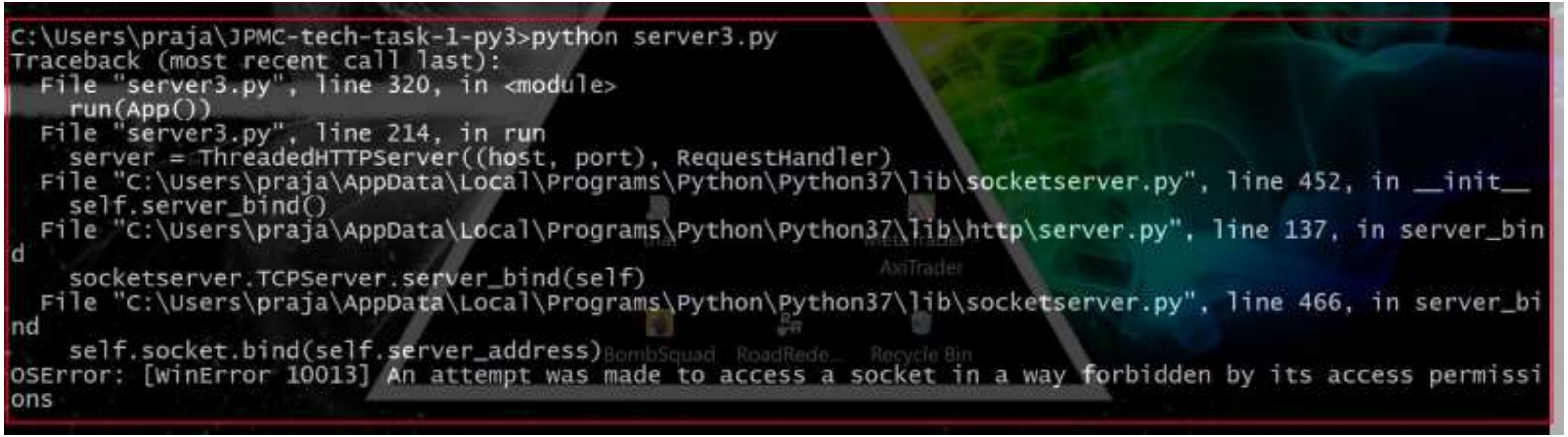
- Then replace the node_modules inside your copy of the repo with the one you downloaded *(make sure the folder is still named node_modules in your repo)*

- Afterwards, you can go ahead and just execute **npm start** (no need to run npm install)

Local Setup: Troubleshooting (Windows)

- In other cases, you might encounter problems running the server app

CASE A:



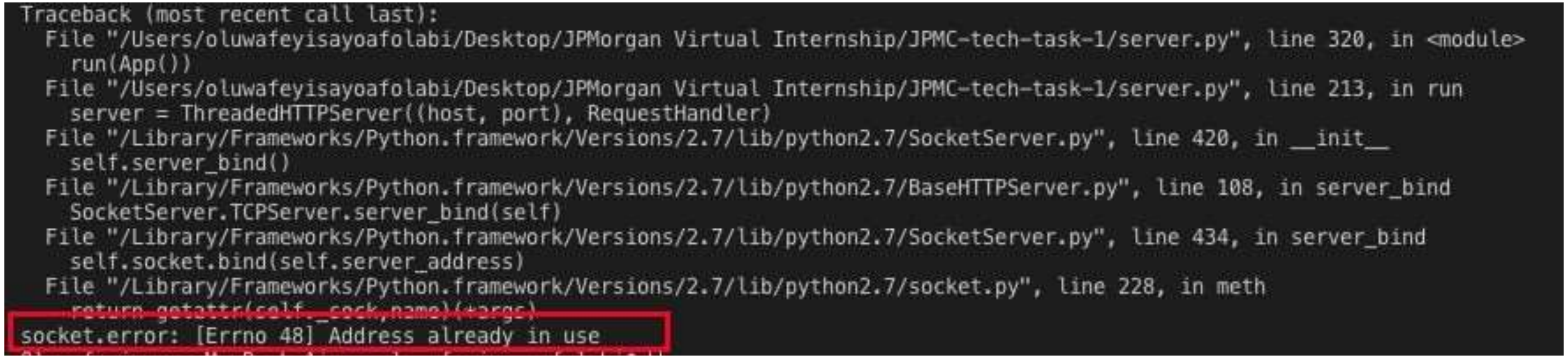
```
C:\Users\praja\JPMC-tech-task-1-py3>python server3.py
Traceback (most recent call last):
  File "server3.py", line 320, in <module>
    run(App())
  File "server3.py", line 214, in run
    server = ThreadedHTTPServer((host, port), RequestHandler)
  File "C:\Users\praja\AppData\Local\Programs\Python\Python37\lib\socketserver.py", line 452, in __init__
    self.server_bind()
  File "C:\Users\praja\AppData\Local\Programs\Python\Python37\lib\http\server.py", line 137, in server_bin
d
    socketserver.TCPServer.server_bind(self)
  File "C:\Users\praja\AppData\Local\Programs\Python\Python37\lib\socketserver.py", line 466, in server_bi
nd
    self.socket.bind(self.server_address)
OSError: [WinError 10013] An attempt was made to access a socket in a way forbidden by its access permissi
ons
```

This is most likely because you have a firewall open preventing you from accessing 8080. You can try the following workarounds:

- Temporarily turn off your firewall
- Using any text editor, open the datafeed/server.py in the repository using your code editor and look for the line where it says port = 8080. change that to port = 8085
- Similarly, open the src/DataStreamer.ts and change the line where it has 8080 to 8085

Local Setup: Troubleshooting (Windows)

CASE B:



```
Traceback (most recent call last):
  File "/Users/oluwafeyisayoafolabi/Desktop/JPMorgan Virtual Internship/JPMC-tech-task-1/server.py", line 320, in <module>
    run(App())
  File "/Users/oluwafeyisayoafolabi/Desktop/JPMorgan Virtual Internship/JPMC-tech-task-1/server.py", line 213, in run
    server = ThreadedHTTPServer((host, port), RequestHandler)
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/SocketServer.py", line 420, in __init__
    self.server_bind()
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/BaseHTTPServer.py", line 108, in server_bind
    SocketServer.TCPServer.server_bind(self)
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/SocketServer.py", line 434, in server_bind
    self.socket.bind(self.server_address)
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/socket.py", line 228, in meth
    return getattr(self._sock,name)(*args)
socket.error: [Errno 48] Address already in use
```

In this case make sure you're only running one instance of the server.py because it hooks itself to port 8080, and once that port is used nothing else can use it. If you want to free that up, terminate the old server.py you're running from one of your terminals by hitting ctrl+c. Alternatively you can kill the process listening on a port (i.e. in this case 8080) by following this guide

Local Setup (Windows)

- If you did encounter any issues, please post your issue/inquiry here: https://github.com/insidesherpa/JPMC-tech-task-3/issues or https://github.com/insidesherpa/JPMC-tech-task-3-py3/issues depending on what repository you chose to work in. When submitting a query, please don't forget to provide as much context as possible, i.e. your OS, what you've done, what your errors is/are, etc (screenshots would help too)

- You can also submit your query in the module page's support modal that pops out when you click the floating element on the page (see image below)

InsideSherpa

JPMorgan Chase Software Engineering Virtual Experience

Software Engineering Task-3

Module 3 – Display data visually for traders

Disclaimer

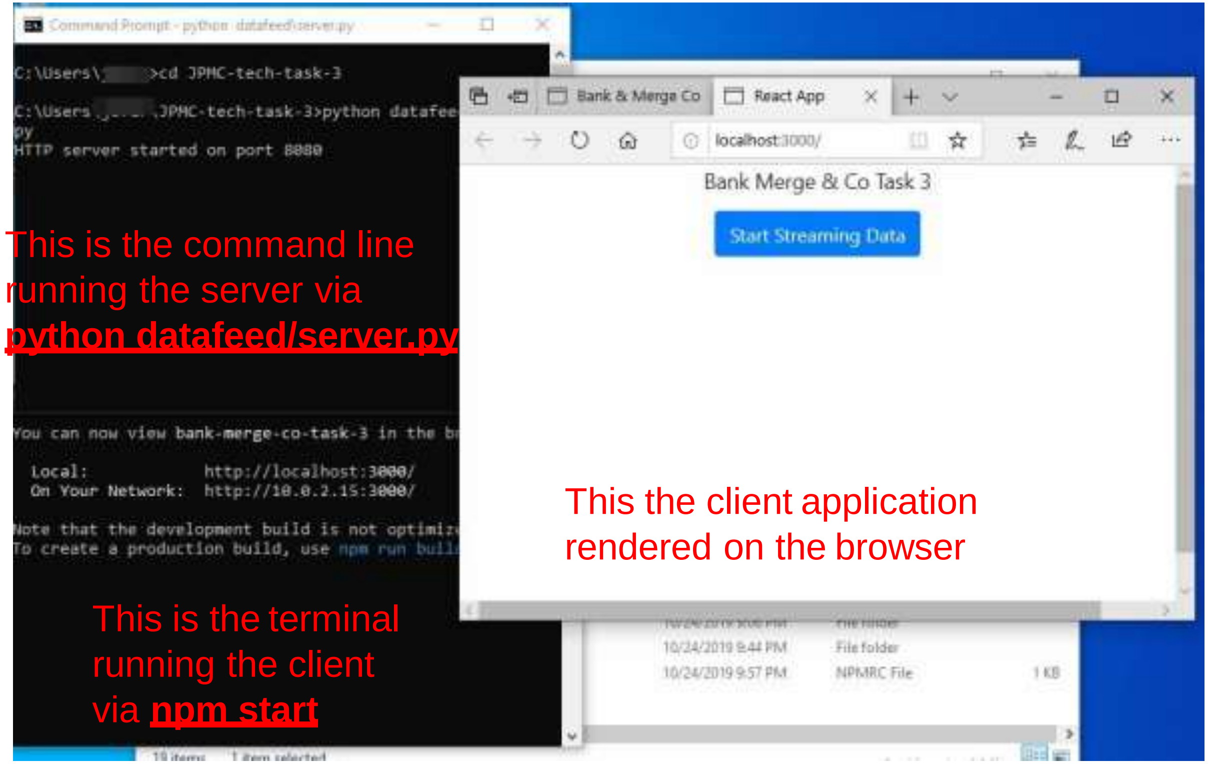
- This guide is only for those who did the setup locally on their machines.

Prerequisite

- Set up should have been done. This means, your server and client applications should have been running with no problems without introducing any changes to the code yet. You can verify this if you get a similar result to any of the following slides that include a picture of the server and client app running together

Prerequisite

Windows OS



```
C:\Users\[redacted]>cd JPMC-tech-task-3

C:\Users\[redacted]\JPMC-tech-task-3>python datafee
py
HTTP server started on port 8080
```

```
You can now view bank-merge-co-task-3 in the br

  Local:            http://localhost:3000/
  On Your Network:  http://10.0.2.15:3000/

Note that the development build is not optimiz
To create a production build, use npm run buil
```

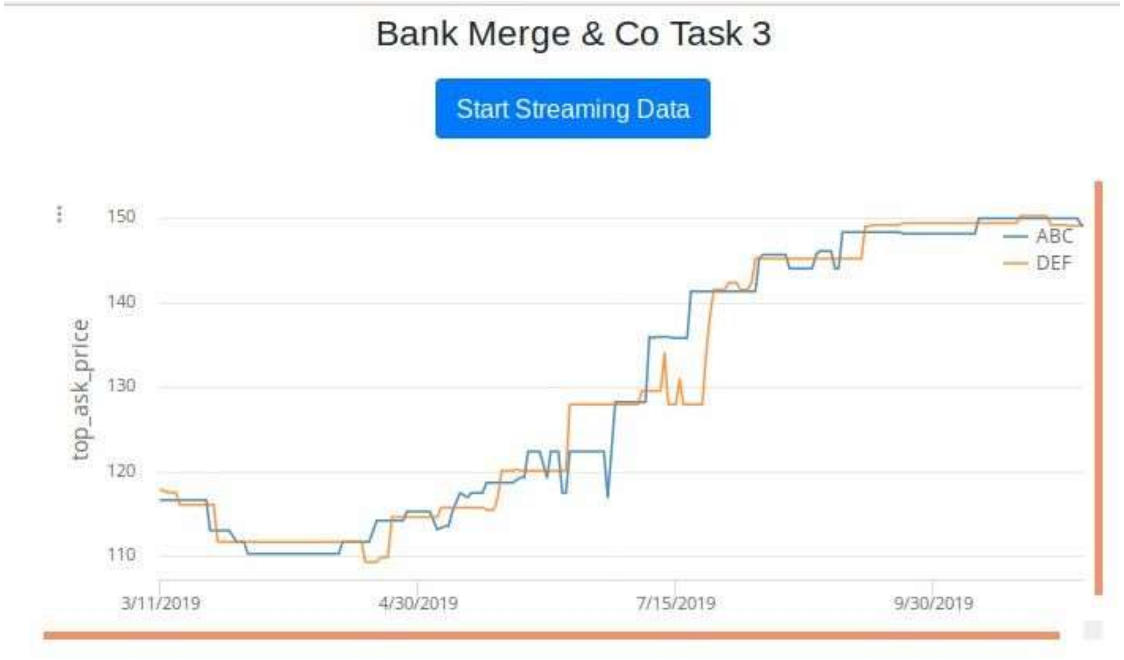
This is the command line running the server via **python datafeed/server.py**

This the client application rendered on the browser

This is the terminal running the client via **npm start**

Note: The state is similar for those who used the python3 version of the repository

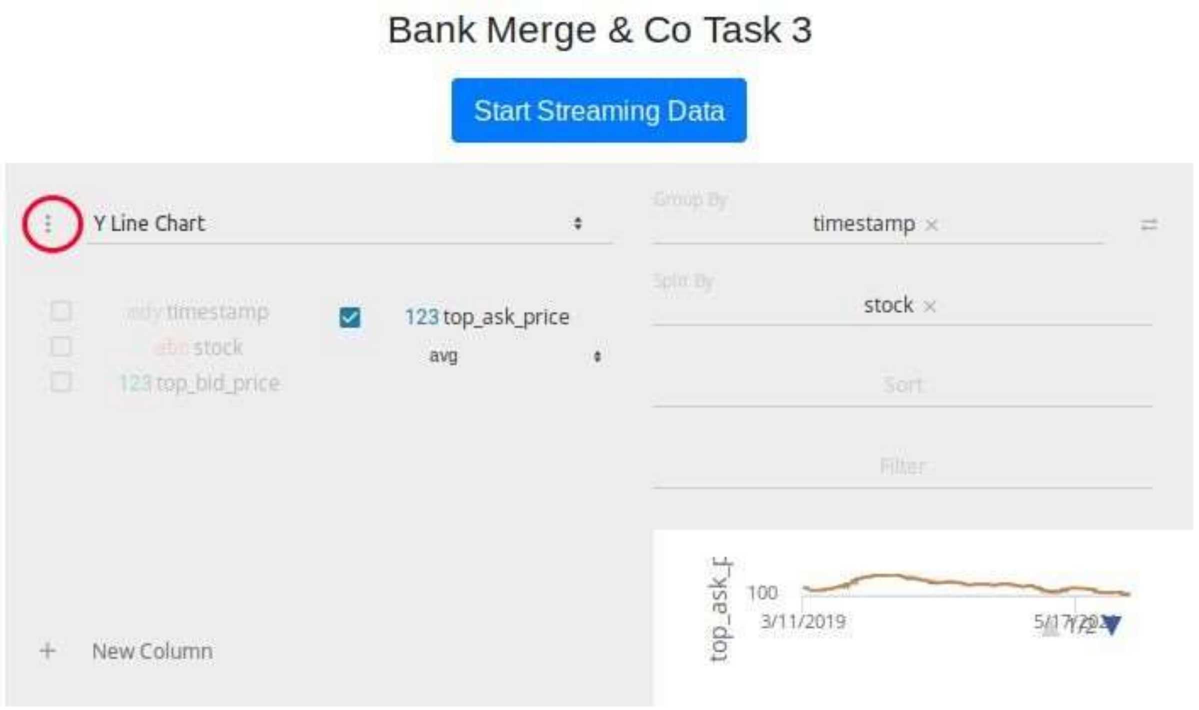
Observe Initial State Of Client App in Browser



This is how the initial state of the client app looks like when you click the blue “**Start Streaming Data”** button

It’s pretty much like the state of task 2 when you’ve finished it. You have two stocks displayed and their top_ask_price changes being tracked through a timeline

Observe Initial State Of Client App in Browser



If you clicked on the 3-dotted button on the upper left corner of the graph you'll see something like image on this slide.

This tells you that the graph is configurable.

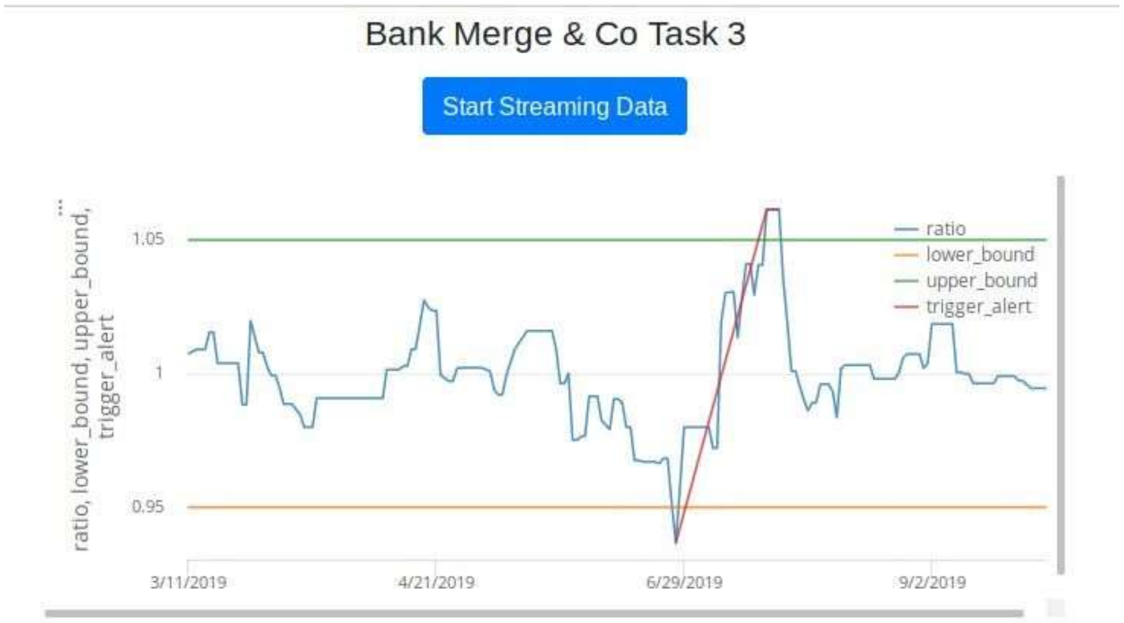
You should know this too by now if you've finished task 2 beforehand

Objectives

- There are two things we have to achieve here to complete this task
  - (1) We now want to make this graph more useful to traders by making it track the **ratio** between two stocks over time and NOT the two stocks' top_ask_price over time.
  - (2) As mentioned before, traders want to monitor the ratio of two stocks against a historical correlation with **upper and lower thresholds/bounds**. This can help them determine a trading opportunity.That said, we also want to make this graph plot those upper and lower thresholds and show when they get crossed by the ratio of the stock

Objectives

- In the end we want to achieve a graph that looks something like this



Objectives

- To achieve this we have to change (2) files: **src/Graph.tsx** and **src/DataManipulator.ts**

- Don’t worry we’ll walk you through how to get these things done

- You can use any text editor your machine has and just open the files in the repository that must be changed (the guide will instruct you in the following slides which files these will be)

- Our recommendation of editors you can use would be VSCode or SublimeText as these are the most commonly used code editors out there.

Making changes in `**Graph.tsx**`

- To accomplish our first objective, we must first make changes to the `**Graph.tsx**` file. Recall, this is the file that takes care of how the Graph component of our App will be rendered and react to the state changes that occur within the App.

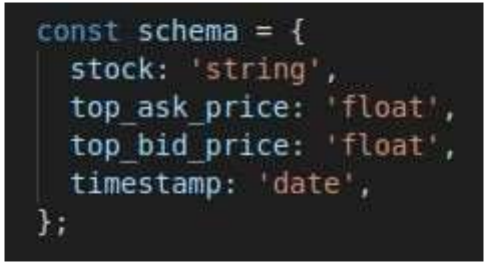
- We're not starting from a static graph anymore and we're basically jumping off from where we've finished in task 2. So the changes we'll be making here aren't going to be as much. What we want to do here is to have one main line tracking the ratio of two stocks and be able to plot upper and lower bounds too.

Making changes in `**Graph.tsx**`

- To do this, we need to modify `**componentDidMount**` method. Recall, the **componentDidMount()** method runs after the component output has been rendered to the DOM. If you want to learn more about it and other lifecycle methods/parts of react components, read more here.

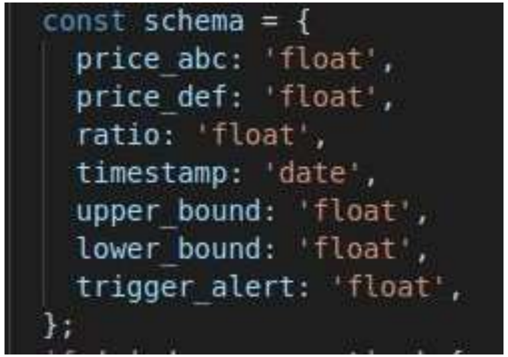
- In this method, we first have to modify the **schema** object as that will dictate how we'll be able to configure the Perspective table view of our graph. In the next slide we'll show how we want to change this exactly

Making changes in `**Graph.tsx**`



```
const schema = {
  stock: 'string',
  top_ask_price: 'float',
  top_bid_price: 'float',
  timestamp: 'date',
};
```

Before



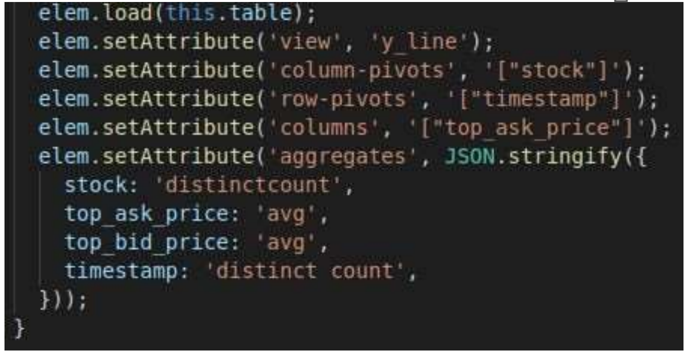
```
const schema = {
  price_abc: 'float',
  price_def: 'float',
  ratio: 'float',
  timestamp: 'date',
  upper_bound: 'float',
  lower_bound: 'float',
  trigger_alert: 'float',
};
```

After

- Notice the changes we’ve made to schema:
  - Since we don’t want to distinguish between two stocks now, but instead want to track their ratios, we made sure to add the **ratio** field. Since we also wanted to track **upper_bound**, **lower_bound,** and the moment when these bounds are crossed i.e. **trigger_alert**, we had to add those fields too.
  - Finally, the reason we added **price_abc** and **price_def** is just because these were necessary to get the **ratio** as you will see later**.** We won’t be configuring the graph to show them anyway.
  - Of course since we’re tracking all of this with respect to time, **timestamp** is going to be there

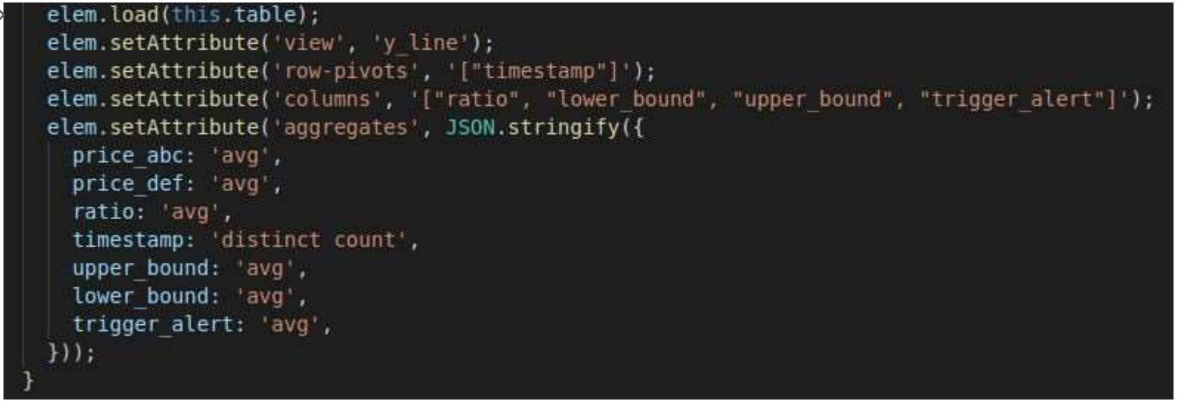
Making changes in `**Graph.tsx**`

- Next, to configure our graph you will need to modify/add more attributes to the element. We've done this before in task 2 so it should be slightly more familiar to you now. (if you forgot you can reread the doc on Perspective configurations particularly on the table.view configurations) The change should look something like:



```
    elem.load(this.table);
    elem.setAttribute('view', 'y_line');
    elem.setAttribute('column-pivots', '["stock"]');
    elem.setAttribute('row-pivots', '["timestamp"]');
    elem.setAttribute('columns', '["top_ask_price"]');
    elem.setAttribute('aggregates', JSON.stringify({
      stock: 'distinctcount',
      top_ask_price: 'avg',
      top_bid_price: 'avg',
      timestamp: 'distinct count',
    }));
}
```

Before



```
    elem.load(this.table);
    elem.setAttribute('view', 'y_line');
    elem.setAttribute('row-pivots', '["timestamp"]');
    elem.setAttribute('columns', '["ratio", "lower_bound", "upper_bound", "trigger_alert"]');
    elem.setAttribute('aggregates', JSON.stringify({
      price_abc: 'avg',
      price_def: 'avg',
      ratio: 'avg',
      timestamp: 'distinct count',
      upper_bound: 'avg',
      lower_bound: 'avg',
      trigger_alert: 'avg',
    }));
}
```

After

Making changes in `**Graph.tsx**`

- ‘**view**’ is the the kind of graph we wanted to visualize the data as. Initially, this is already set to **y_line**. This is the type of graph we want so we’re good here.

- **‘column-pivots’** used to exist and was what allowed us to distinguish / split stock ABC with DEF back in task 2. We removed this because we’re concerned about the ratios between two stocks and not their separate prices

- ‘**row-pivots’** takes care of our x-axis. This allows us to map each datapoint based on the timestamp it has. Without this, the x-axis is blank. So this field and its value remains

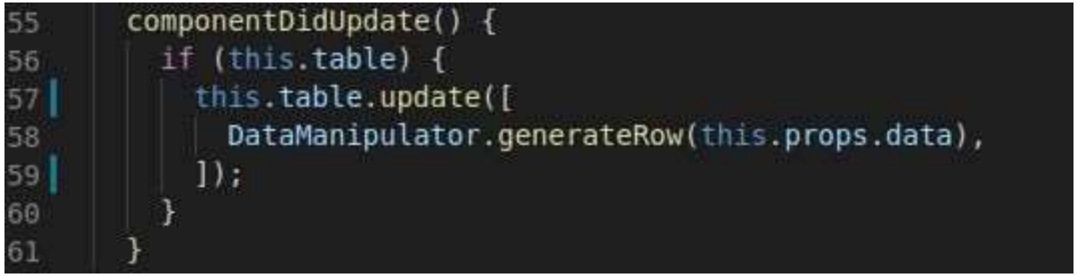
Making changes in **`Graph.tsx`**

- ‘**columns’** is what will allow us to only focus on a particular part of a datapoint’s data along the y-axis. Without this, the graph will plot all the fields and values of each datapoint and it will be a lot of noise. For this case, we want to track **ratio, lower_bound, upper_bound** and **trigger_alert**.

- **‘aggregates’** is what will allow us to handle the cases of duplicated data we observed way back in task 2 and consolidate them as just one data point. In our case we only want to consider a data point unique if it has a timestamp. Otherwise, we will average out the all the values of the other non-unique fields these ‘similar’ datapoints before treating them as one (e.g. ratio, price_abc, …)

Making changes in `**Graph.tsx**`

- Finally, we have to make a slight update in the **componentDidUpdate** method. This method is another component lifecycle method that gets executed whenever the component updates, i.e. when the graph gets updated in our case. The change we want to make is on the argument we put in **this.table.update.** This is how it's supposed to look like after the change:



```
componentDidUpdate() {
  if (this.table) {
    this.table.update([
      DataManipulator.generateRow(this.props.data),
    ]);
  }
}
```

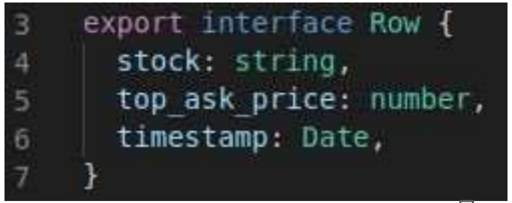
There's a reason why we did this change but you'll understand it in the next couple of slides, based on our changes in **DataManipulator.ts**

Making changes in `**DataManipulator.ts**`

- To fully achieve our goal in this task, we have to make some modifications in the **DataManipulator.ts** file. This file will be responsible for processing the raw stock data we've received from the server before it throws it back to the Graph component's table to render. Initially, it's not really doing any processing hence we were able to keep the status quo from the finished product in task 2

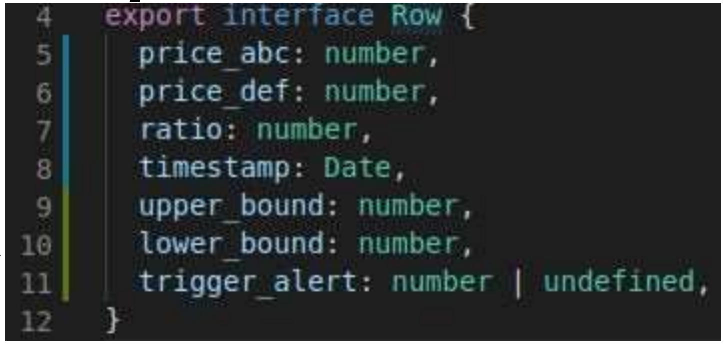
- The first thing we have to modify in this file is the **Row** interface. If you notice, the initial setting of the **Row** interface is almost the same as the old **schema** in **Graph.tsx** before we updated it. So now, we have to update it to match the new **schema**. See next slide to better visualize the change that's supposed to happen.

Making changes in `**DataManipulator.ts**`



```
export interface Row {
  stock: string,
  top_ask_price: number,
  timestamp: Date,
}
```

Before



```
export interface Row {
  price_abc: number,
  price_def: number,
  ratio: number,
  timestamp: Date,
  upper_bound: number,
  lower_bound: number,
  trigger_alert: number | undefined,
}
```

After

- This change is necessary because it will be the structure of the return object of the only function of the DataManipulator class, i.e. the **generateRow** function
- It's important that the return object corresponds to the the **schema** of the table we'll be updating in the Graph component because that's the only way that we'll be able to display the right output we want.

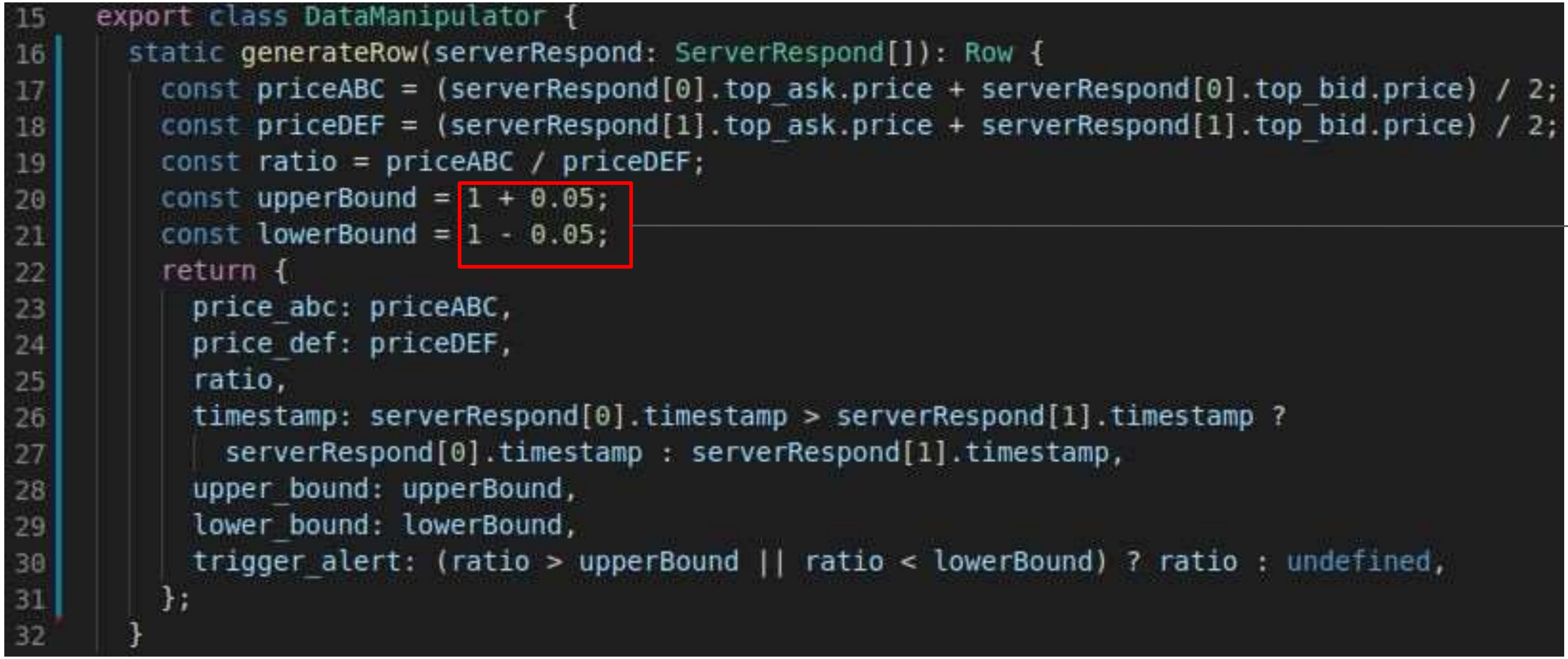
note: Interfaces help define the values a certain entity must have. If you want to learn more about interfaces in Typescript you can read this material in your spare time

Making changes in **`DataManipulator.ts`**

- Finally, we have to update the **generateRow** function of the DataManipulator class to properly process the raw server data passed to it so that it can return the processed data which will be rendered by the Graph component's table.

- Here we can compute for **price_abc** and **price_def** properly (like what you did back in task 1). Afterwards we can also compute for **ratio** using the two computed prices, (like what you did in task 1 too). And, set **lower** and **upper** bounds, as well as **trigger_alert**. To better understand this see the expected change in the next slide

Making changes in `**DataManipulator.ts**`



```
export class DataManipulator {
  static generateRow(serverRespond: ServerRespond[]): Row {
    const priceABC = (serverRespond[0].top_ask.price + serverRespond[0].top_bid.price) / 2;
    const priceDEF = (serverRespond[1].top_ask.price + serverRespond[1].top_bid.price) / 2;
    const ratio = priceABC / priceDEF;
    const upperBound = 1 + 0.05;
    const lowerBound = 1 - 0.05;
    return {
      price_abc: priceABC,
      price_def: priceDEF,
      ratio,
      timestamp: serverRespond[0].timestamp > serverRespond[1].timestamp ?
        serverRespond[0].timestamp : serverRespond[1].timestamp,
      upper_bound: upperBound,
      lower_bound: lowerBound,
      trigger_alert: (ratio > upperBound || ratio < lowerBound) ? ratio : undefined,
    };
  }
```

Feel free to change this to +/-10% of the 12 month historical average ratio

This was just for a test value

Making changes in `**DataManipulator.ts**`

- Observe how we’re able to access **serverRespond** as an array where in the first element (0-index) is about stock ABC and the second element (1-index) is about stock DEF. With this, we were able to easily just plug in values to the formulas we used back in task 1 to compute for prices and ratio properly

- Also note how the return value is changed from an array of **Row objects** to just a single **Row object** This change explains why we also adjusted the argument we passed to **table.update** in Graph.tsx earlier so that consistency is preserved.

Making changes in `**DataManipulator.ts**`

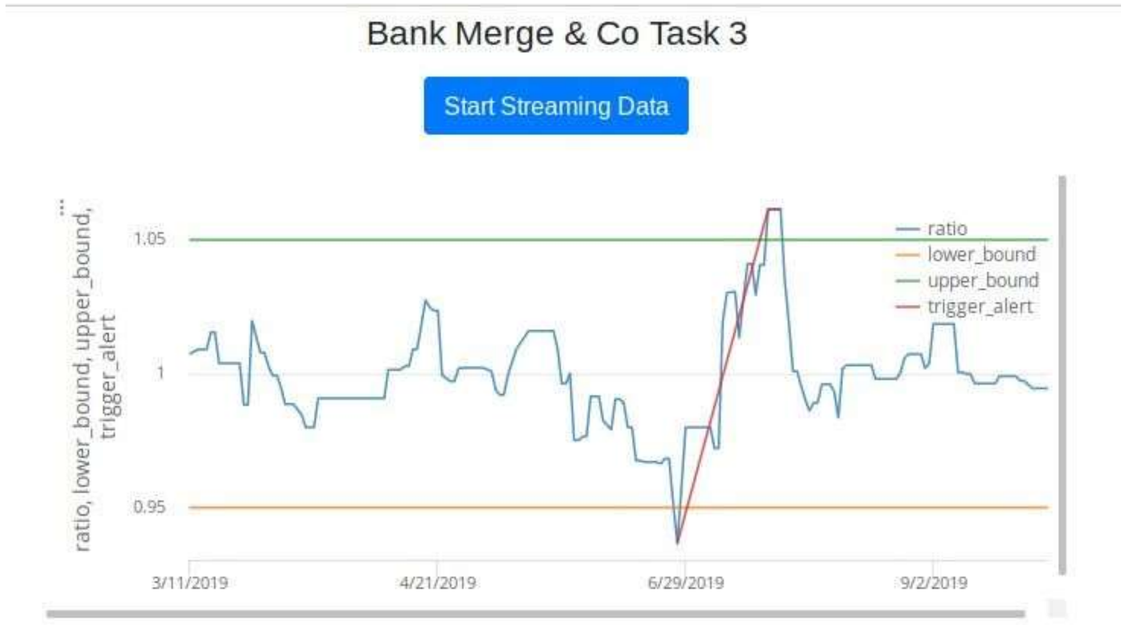
- The **upper_bound** and **lower_bound** are pretty much constant for any data point. This is how we will be able to maintain them as steady upper and lower lines in the graph. While 1.05 and 0.95 isn’t really +/-10% of the 12 month historical average ratio (i.e. 1.1 and 0.99) you’re free to play around with the values and see which has a more conservative alerting behavior.

- The **trigger_alert** field is pretty much just a field that has a value (e.g. the ratio) if the threshold is passed by the ratio. Otherwise if the ratio remains within the threshold, then no value/undefined will suffice.

Wrapping up

- Changes in **Graph.tsx** and **DataManipulator.ts** are done.
- By now you should’ve accomplished all the objectives of the task
- Feel free to poke around before completely saving everything and creating your patch file, e.g. see the different effects of changing the configurations would do to your table/graph.
- Please don’t forget to leave comments in your code especially at the places where the fixes for bugs and where you piped in the data feed - this will help with other team member's understanding of your work.

End Result


Bank Merge & Co Task 3
Start Streaming Data
ratio, lower_bound, upper_bound, trigger_alert
1.05
1
0.95
3/11/2019
4/21/2019
6/29/2019
9/2/2019
ratio
lower_bound
upper_bound
trigger_alert

Conclusion

I participated in JPMorgan Chase’s virtual job simulation on the Forage platform, and it was incredibly useful to understand what it might be like to participate in a software engineering team at JPMorgan Chase.

I had the opportunity to use JPMorgan’s own open source library called Perspective, to set up my dev environment and to build a live graph that displays a data feed for traders to monitor. It helped me build gain experience coding in a realistic context and even provided me with the opportunity to make an open source contribution at the end.

Through this program I realized that I really enjoy working on building technical tools that are used by traders and would love to apply what I've learned in a software engineering team at a company like JPMorgan Chase.

InsideSherpa

The End