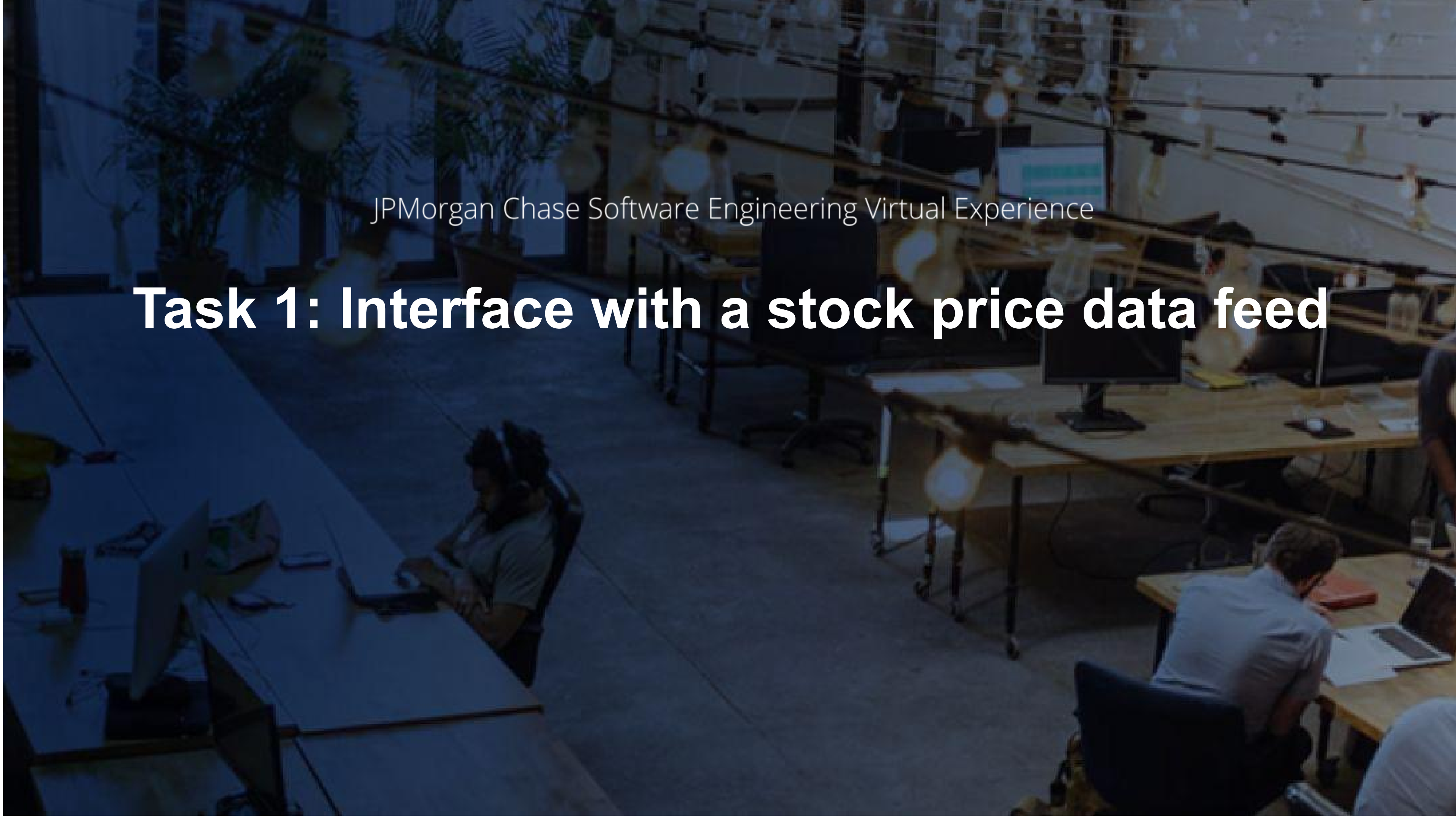


JPMorgan Chase Software Engineering Virtual Experience

Task 1: Interface with a stock price data feed

First Step

- Now that you have your local development environment set up, it’s time to actually run the program. Run server.py and client.py (in that order) and take a look at the output. You should notice two things:
    - (1) The Ratio in the client output is always 1
    - (2) The price of each stock is always the same as its bid

- These are obviously wrong so your job is to fix them…

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

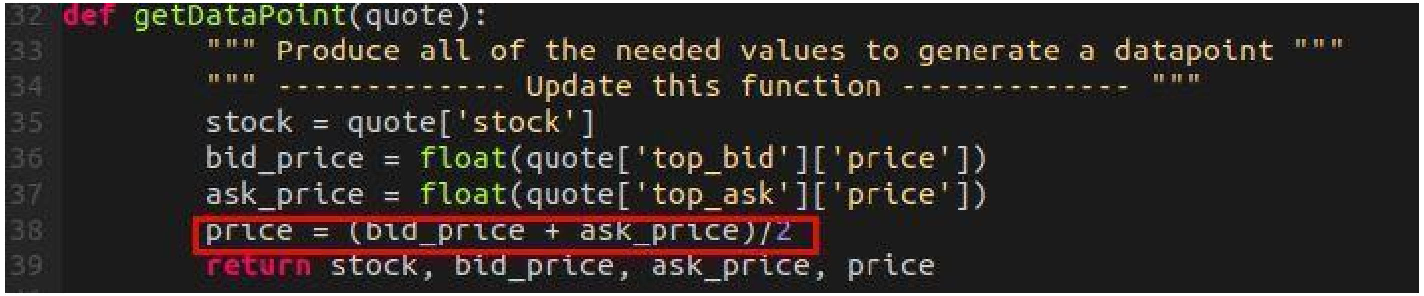
- The changes you need to make will be in the following methods of the file
  - getDataPoint
  - getRatio
  - Main

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

Making changes in client.py **getDataPoint**

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

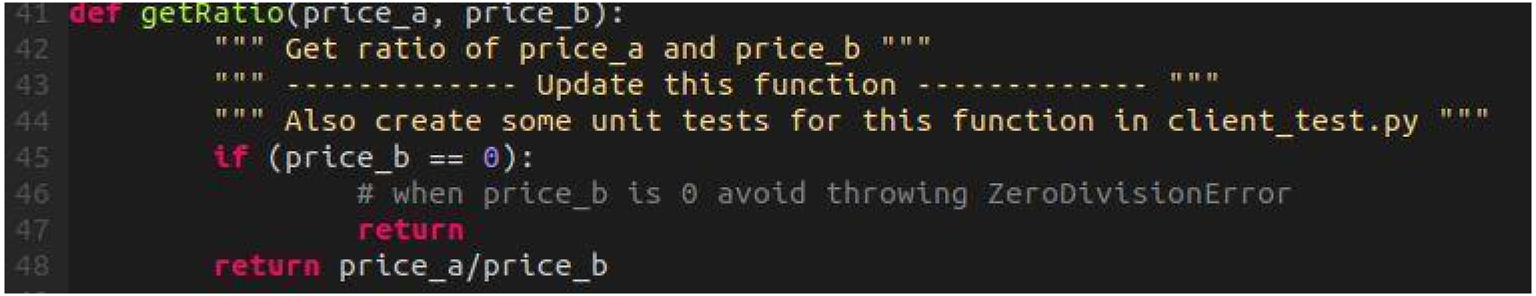
YOU DO NOT NEED TO CHANGE the return value as that is representational of the entire data point. You should end up with something like:



```
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
getRatio

getRatio. Right now, 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**



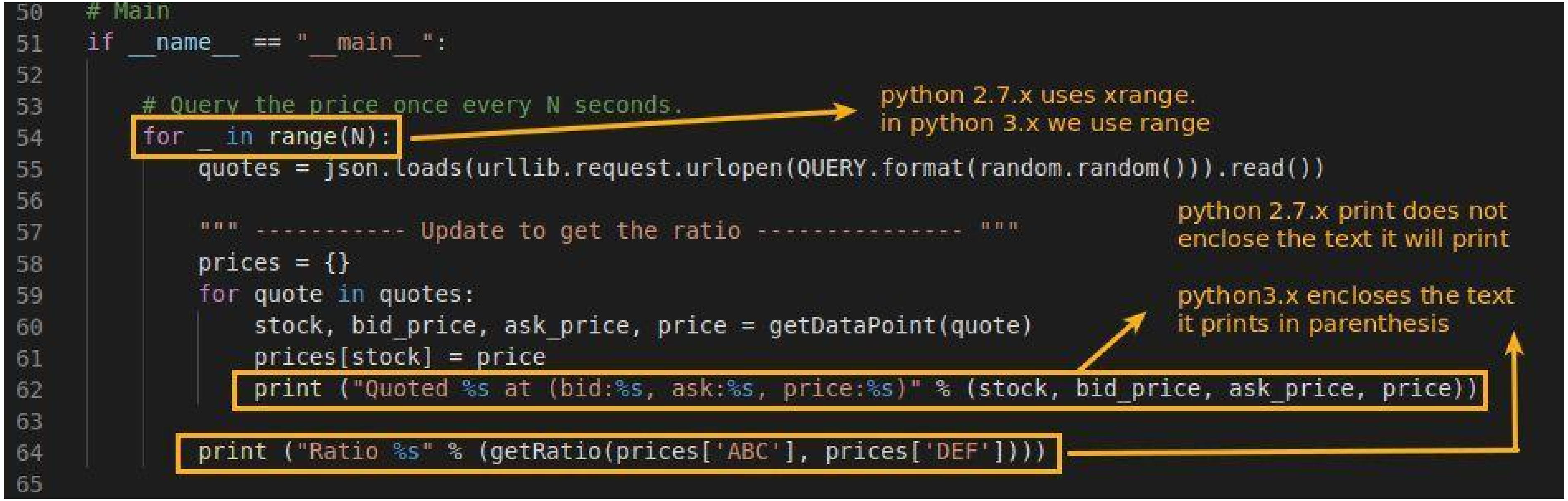
```
def getRatio(price_a, price_b):
        """ Get ratio of price_a and price_b """
        """ ------------- Update this function ------------- """
        """ Also create some unit tests for this function in client_test.py """
        if (price_b == 0):
                # when price_b is 0 avoid throwing 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
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 the getDataPoint method so that you can properly call getRatio and print the right ratio.

Making changes in client.py
main



```
# 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
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
main

- To review, we created 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, the 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.