Graphical user interface for interacting with IEX REST API built with PyQt

Eric Stevens
June 6, 2019

The Goal

REST - IEX

- Build library code to accomplish specific tasks (retrieve specific data sets) from IEX using the IEX REST API.

GUI - PyQt5

- Build a user interface that allows API calls to be made graphically, without the need for the user to write any code.

Motivation

Professional

- Experience with in-demand skill (REST API interaction)
- Portfolio optimization (Theory: GUI > then bunch of random code)

Ethical

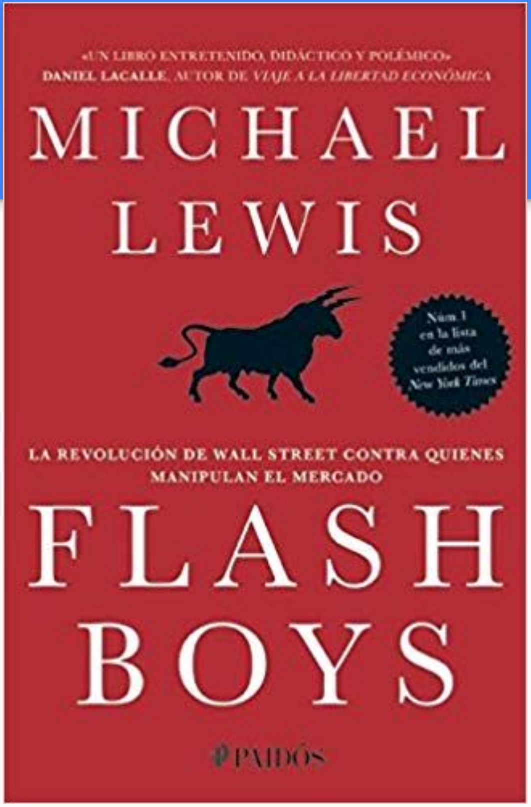
- IEX is the solution to a major problem in the stock market. Working closely with it makes me feel like I am doing something good.

Background

Flash Boys

This project was inspired by the book "Flash Boys" by Michael Lewis.

Exposes the perversion of capitalism that resulted from modern technology called "latency arbitrage" or "electronic front running" which you may have heard referred to as high frequency trading (HFT).


«UN LIBRO ENTRETENIDO, DIDÁCTICO Y POLÉMICO»
DANIEL LACALLE, AUTOR DE *VIAJE A LA LIBERTAD ECONÓMICA*

MICHAEL LEWIS

Núm. 1 en la lista de más vendidos del *New York Times*

LA REVOLUCIÓN DE WALL STREET CONTRA QUIENES MANIPULAN EL MERCADO

FLASH BOYS

PAIDÓS

Background

Brad Katsuyama

Good man at the right place and time.

Tasked by Royal Bank of Canada (RBC) to lead electronic products team.

In the process, he untangled the mysteries of the problems in the stock market due to the HFT underworld and built a solution.

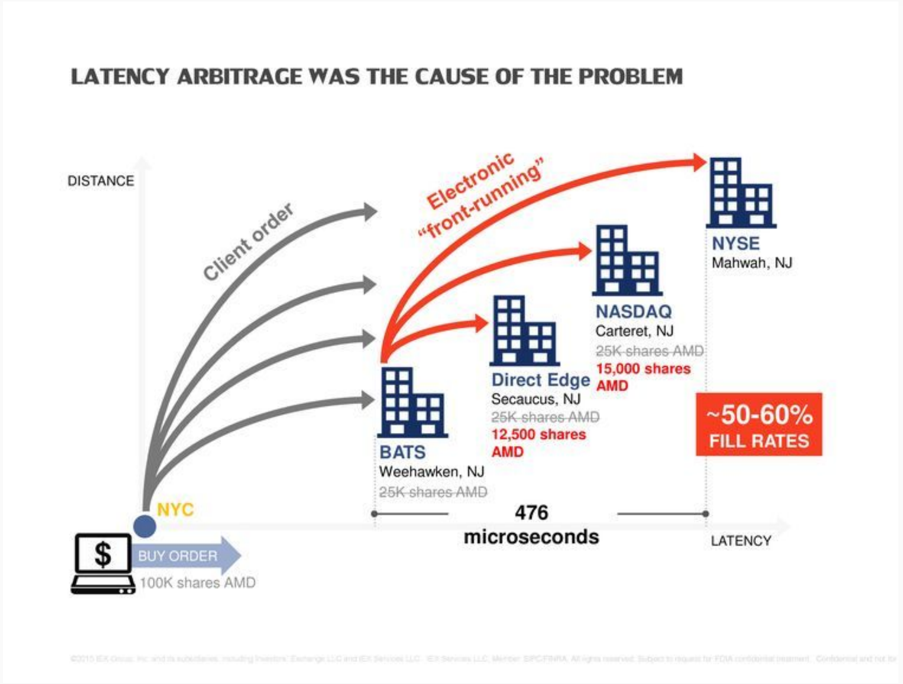
https://www.businessinsider.com/flash-boys-star-iex-ceo-brad-katsuyama-fighting-duopoly-2018-9

Background

The Problem

Clients see an available number of shares for sale at a price. They attempt to by all the shares available at the visible price. Only a fraction of the orders are filled. Orders arrive at different exchanges at different times. HFT firms see orders and buy up other shares at other exchanges.

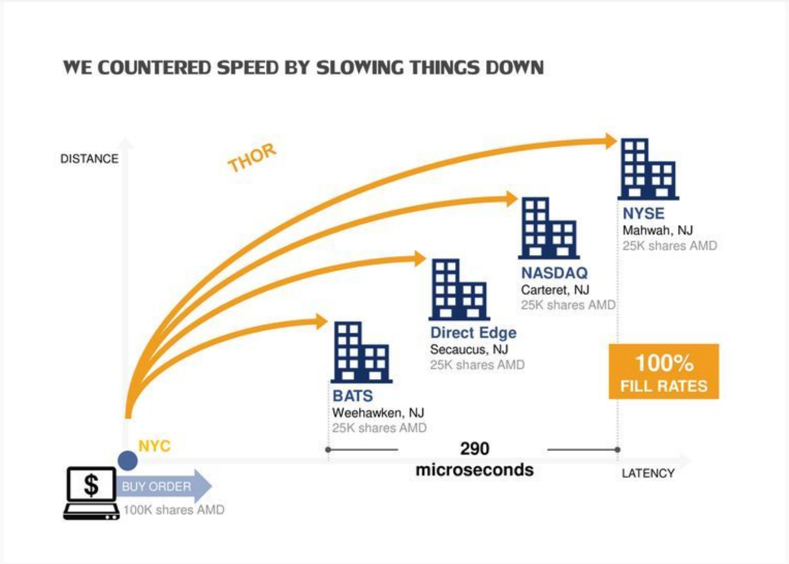
https://slideplayer.com/slide/14353054/



Background

A Solution

Working with RBC, Mr. Katsuyama and his team invented a networking product that would slow electronic orders down based on the relative distance to the exchanges. This ment all orders would reach their target exchanges at the same time, eliminating the possibility of front running.



https://slideplayer.com/slide/14353054/

TO THIS DAY STOCK EXCHANGES MAKE MORE MONEY CATERING TO BROKERAGES AND HFTs THAN THEY DO EXECUTING ORDERS.

EXCHANGES PAY BROKERS TO ROUTE ORDERS TO THEM IN ORDER TO GET PAID BY HFTs FOR SPECIAL ACCESS.

Background

IEX - The Investors Exchange

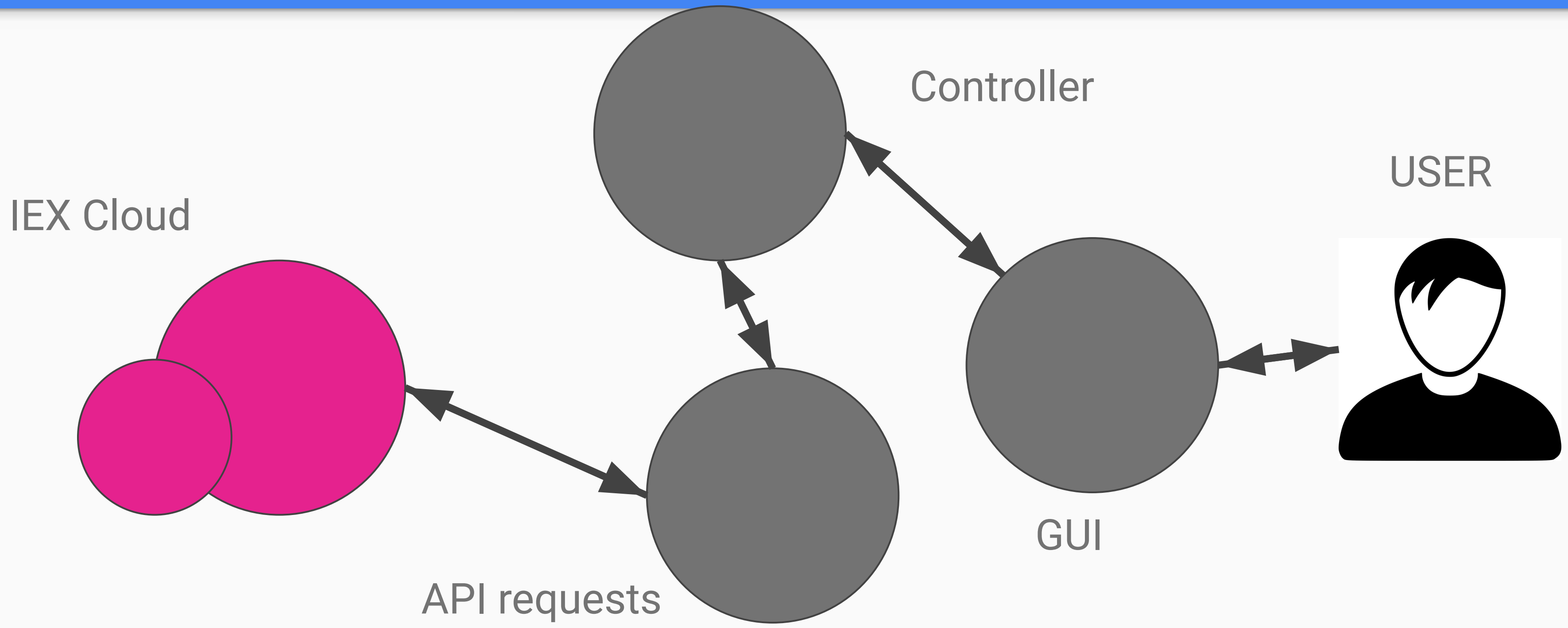
Stock exchanges had become corrupt. They were making more money selling data products to HFT firms than actually executing orders.

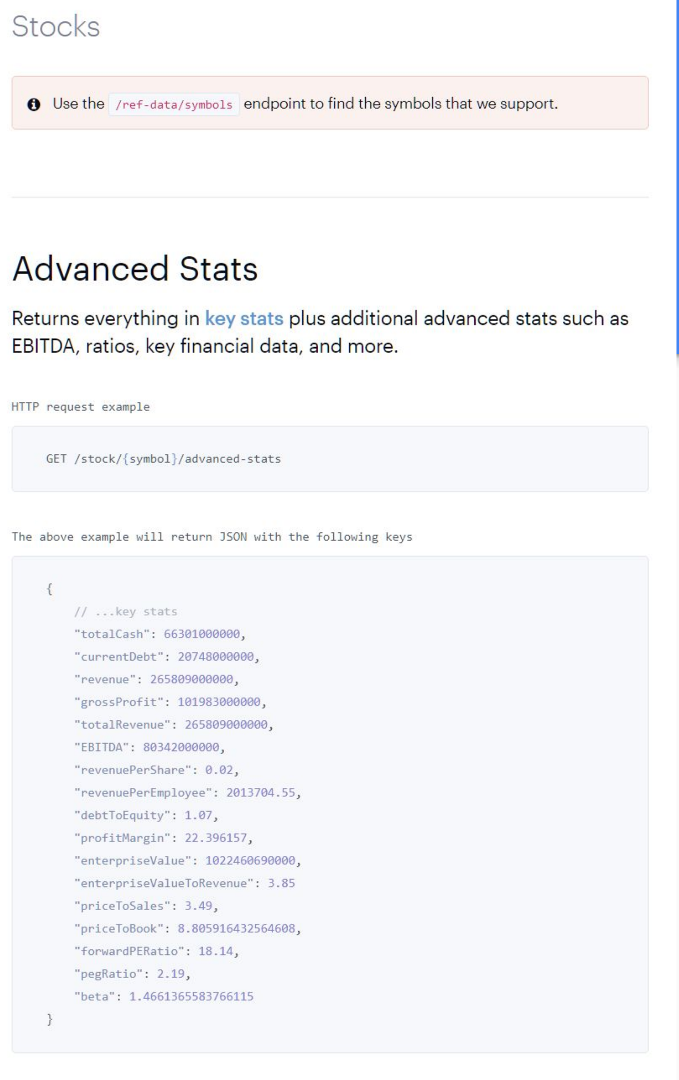
Brad Katsuyama started a new exchange that would shield its customers from the types of predators that other exchanges were catering to.

The Project

- **PyQt5** - PyQt5 is the library that is used to build out the UI
- **Requests** - requests is a library that handles networking, allowing users to make http requests in python. It returns the status of a request, the content of the request, and has the ability to take returned json objects and convert them to lists of dictionaries.
- **Json** - The json library is useful for dumping dictionaries. It is used in places through the code to turn dictionaries to text.

Architecture (somewhat MVC like)





Stocks

Use the `/ref-data/symbols` endpoint to find the symbols that we support.

Advanced Stats

Returns everything in key stats plus additional advanced stats such as EBITDA, ratios, key financial data, and more.

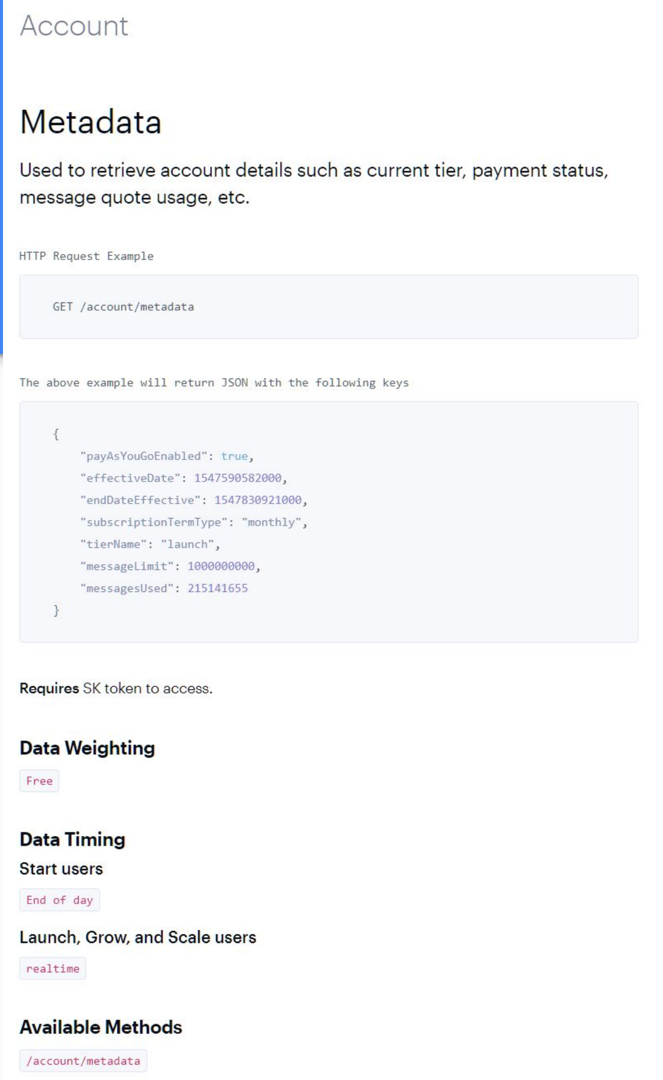
HTTP request example

```
GET /stock/{symbol}/advanced-stats
```

The above example will return JSON with the following keys

```
{
    // ...key stats
    "totalCash": 66301000000,
    "currentDebt": 20748000000,
    "revenue": 265809000000,
    "grossProfit": 101983000000,
    "totalRevenue": 265809000000,
    "EBITDA": 80342000000,
    "revenuePerShare": 0.02,
    "revenuePerEmployee": 2013704.55,
    "debtToEquity": 1.07,
    "profitMargin": 22.396157,
    "enterpriseValue": 1022460690000,
    "enterpriseValueToRevenue": 3.85
    "priceToSales": 3.49,
    "priceToBook": 8.805916432564608,
    "forwardPERatio": 18.14,
    "pegRatio": 2.19,
    "beta": 1.4661365583766115
}
```

IEX Cloud



Account

Metadata

Used to retrieve account details such as current tier, payment status, message quote usage, etc.

HTTP Request Example

```
GET /account/metadata
```

The above example will return JSON with the following keys

```
{
    "payAsYouGoEnabled": true,
    "effectiveDate": 1547590582000,
    "endDateEffective": 1547830921000,
    "subscriptionTermType": "monthly",
    "tierName": "launch",
    "messageLimit": 1000000000,
    "messagesUsed": 215141655
}
```

Requires SK token to access.

Data Weighting

`Free`

Data Timing

Start users

`End of day`

Launch, Grow, and Scale users

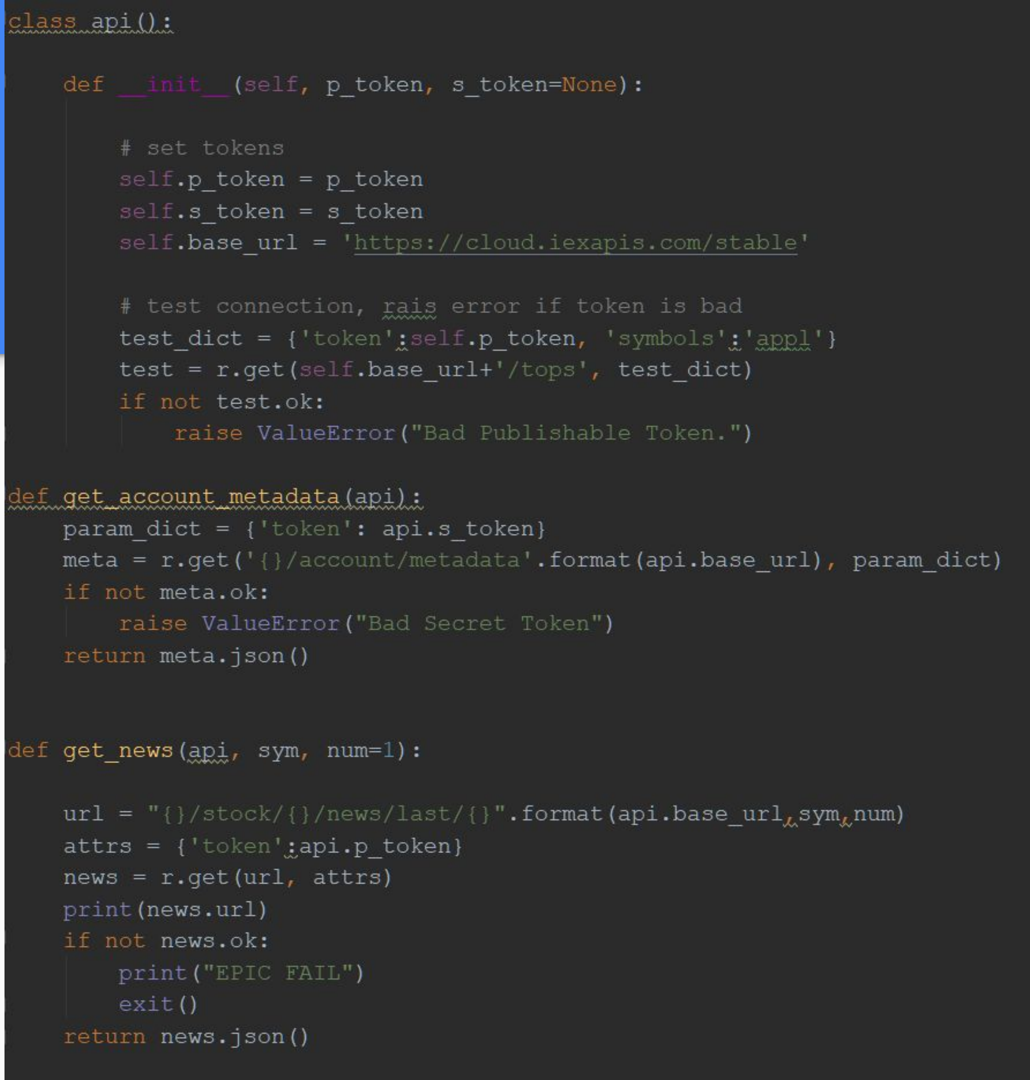
`realtime`

Available Methods

`/account/metadata`

API Request Library

- **Api class:** This object holds user authentication tokens used in every api call.
- Other functions take in an api() class object and receive data defined by the function.



```python
class api():

    def __init__(self, p_token, s_token=None):

        # set tokens
        self.p_token = p_token
        self.s_token = s_token
        self.base_url = 'https://cloud.iexapis.com/stable'

        # test connection, rais error if token is bad
        test_dict = {'token':self.p_token, 'symbols':'appl'}
        test = r.get(self.base_url+'/tops', test_dict)
        if not test.ok:
            raise ValueError("Bad Publishable Token.")

def get_account_metadata(api):
    param_dict = {'token': api.s_token}
    meta = r.get('{}/account/metadata'.format(api.base_url), param_dict)
    if not meta.ok:
        raise ValueError("Bad Secret Token")
    return meta.json()


def get_news(api, sym, num=1):

    url = "{}/stock/{}/news/last/{}".format(api.base_url,sym,num)
    attrs = {'token':api.p_token}
    news = r.get(url, attrs)
    print(news.url)
    if not news.ok:
        print("EPIC FAIL")
        exit()
    return news.json()
```

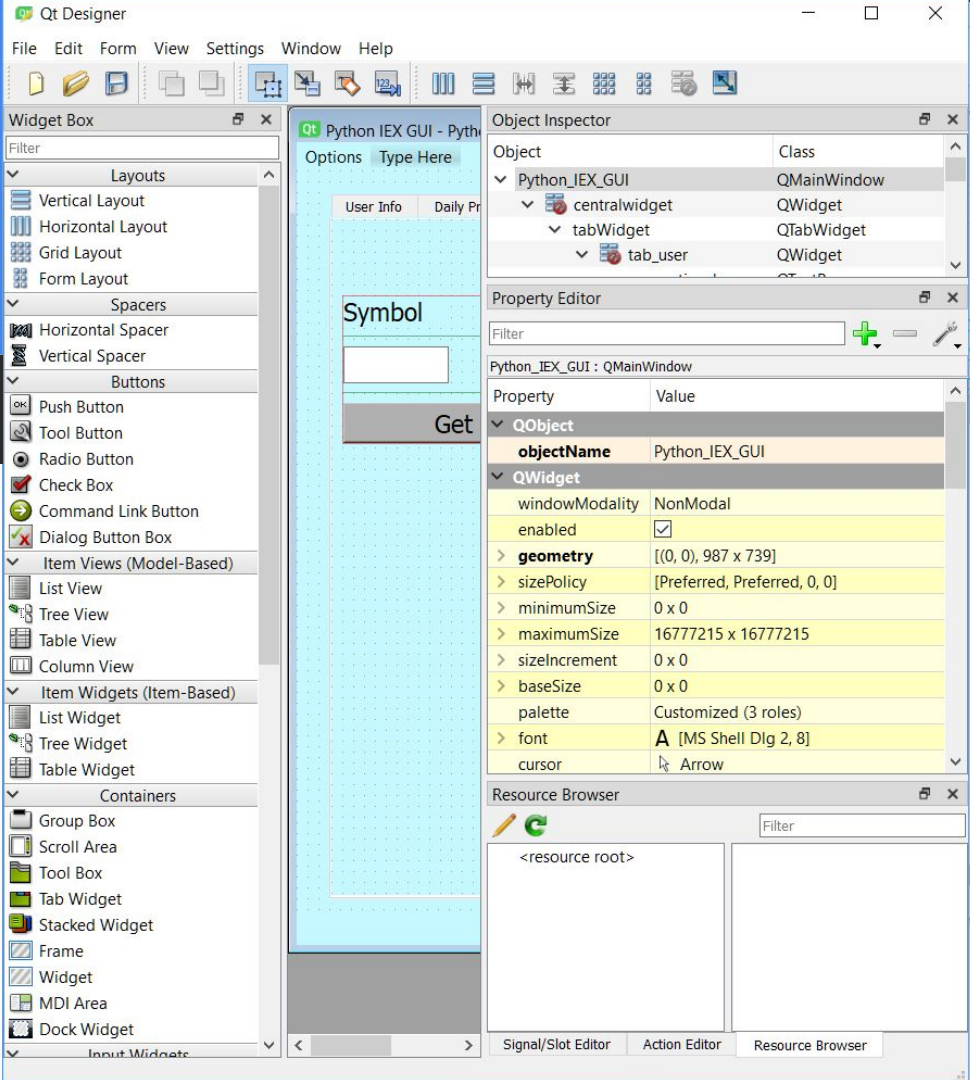
GUI - Qt Designer

```
import os
os.system(r"pyuic5 ui_files/Python_IEX_GUI.ui > Python_IEX_GUI.py")
os.system(r"python ui_files/MainGUI.py")
```

GUI for building GUIs

Allows styling and component naming on the fly.

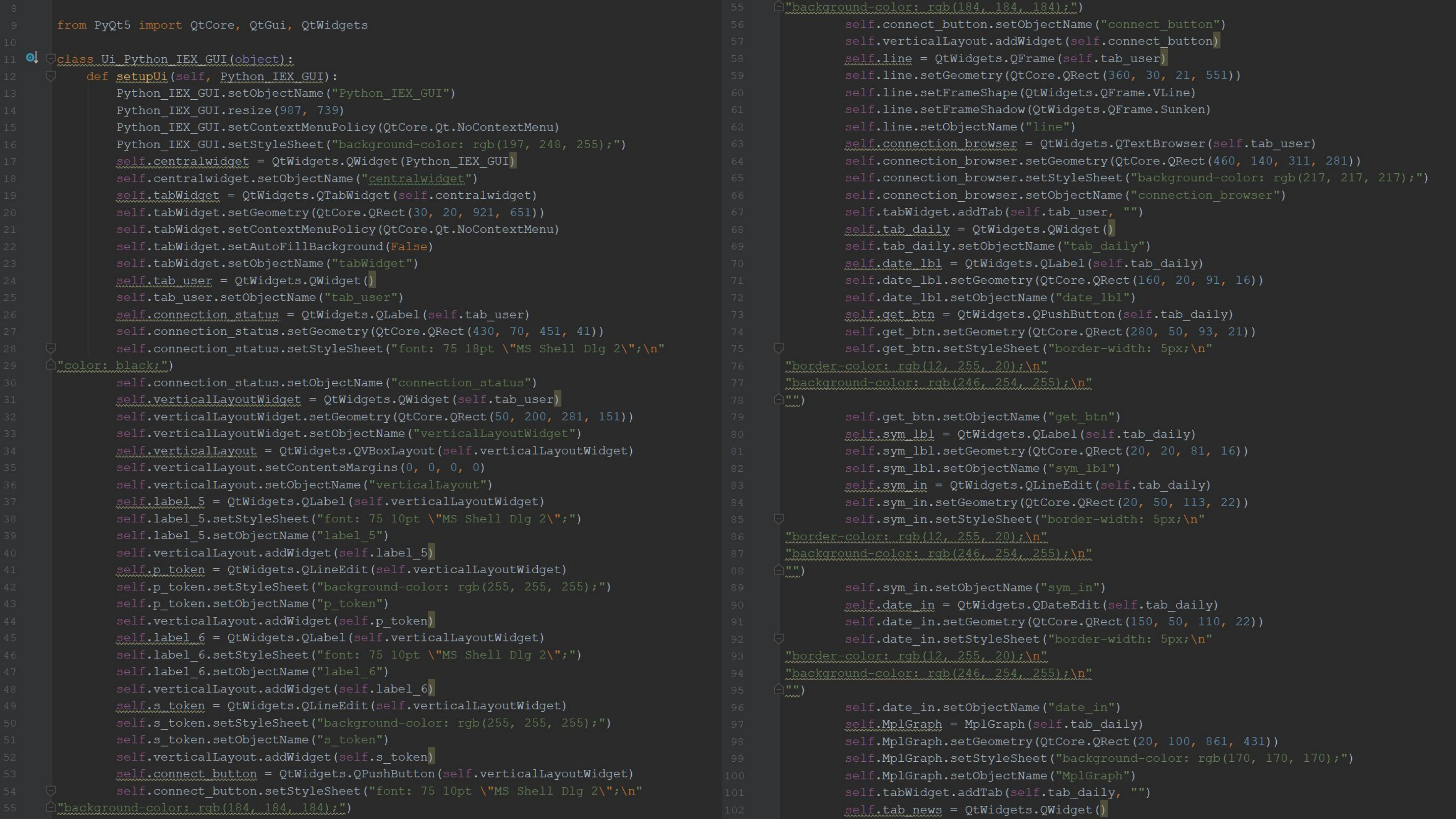
The little script at the top allowed me to see changes immediately



Object	Class
Python_IEX_GUI	QMainWindow
centralwidget	QWidget
tabWidget	QTabWidget
tab_user	QWidget

Python_IEX_GUI : QMainWindow

Property	Value
QObject	
objectName	Python_IEX_GUI
QWidget	
windowModality	NonModal
enabled	✓
geometry	[(0, 0), 987 x 739]
sizePolicy	[Preferred, Preferred, 0, 0]
minimumSize	0 x 0
maximumSize	16777215 x 16777215
sizeIncrement	0 x 0
baseSize	0 x 0
palette	Customized (3 roles)
font	A [MS Shell Dlg 2, 8]
cursor	Arrow



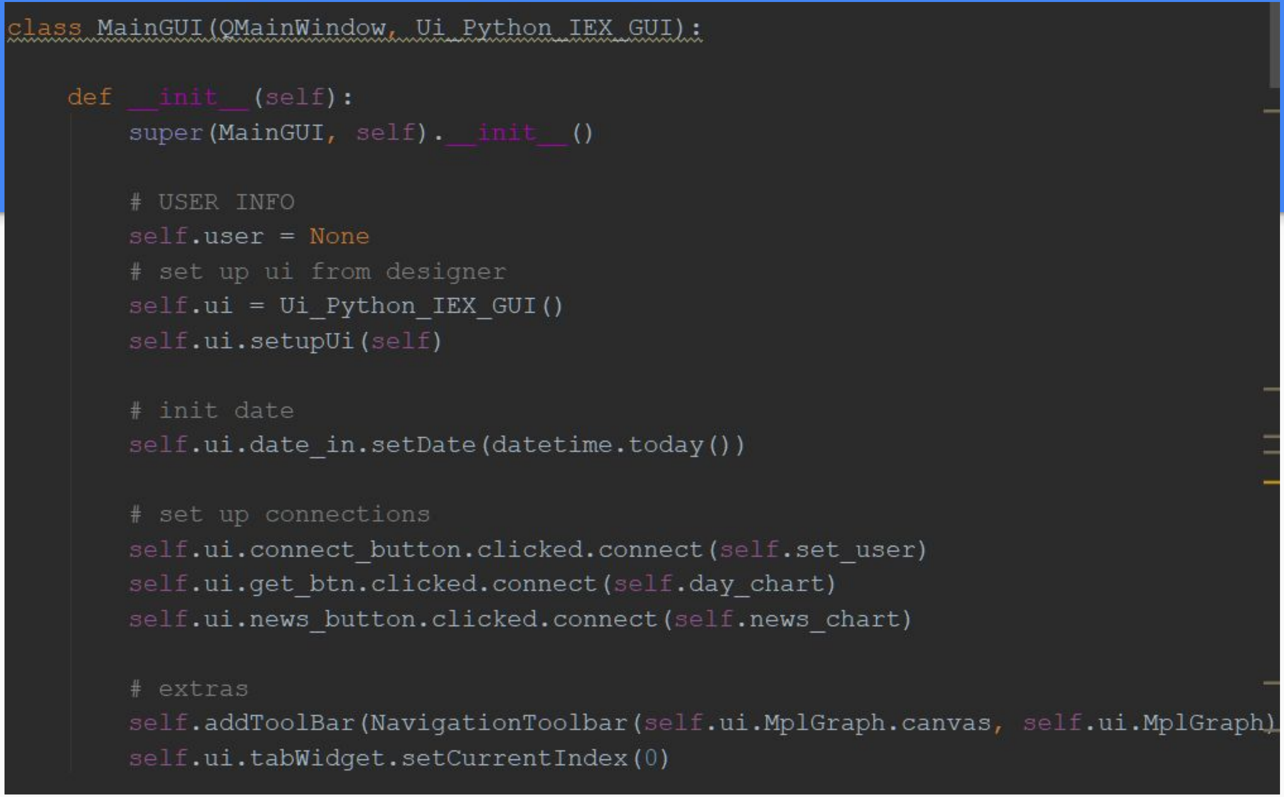
```python
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_Python_IEX_GUI(object):
    def setupUi(self, Python_IEX_GUI):
        Python_IEX_GUI.setObjectName("Python_IEX_GUI")
        Python_IEX_GUI.resize(987, 739)
        Python_IEX_GUI.setContextMenuPolicy(QtCore.Qt.NoContextMenu)
        Python_IEX_GUI.setStyleSheet("background-color: rgb(197, 248, 255);")
        self.centralwidget = QtWidgets.QWidget(Python_IEX_GUI)
        self.centralwidget.setObjectName("centralwidget")
        self.tabWidget = QtWidgets.QTabWidget(self.centralwidget)
        self.tabWidget.setGeometry(QtCore.QRect(30, 20, 921, 651))
        self.tabWidget.setContextMenuPolicy(QtCore.Qt.NoContextMenu)
        self.tabWidget.setAutoFillBackground(False)
        self.tabWidget.setObjectName("tabWidget")
        self.tab_user = QtWidgets.QWidget()
        self.tab_user.setObjectName("tab_user")
        self.connection_status = QtWidgets.QLabel(self.tab_user)
        self.connection_status.setGeometry(QtCore.QRect(430, 70, 451, 41))
        self.connection_status.setStyleSheet("font: 75 18pt \"MS Shell Dlg 2\";\n"
"color: black;")
        self.connection_status.setObjectName("connection_status")
        self.verticalLayoutWidget = QtWidgets.QWidget(self.tab_user)
        self.verticalLayoutWidget.setGeometry(QtCore.QRect(50, 200, 281, 151))
        self.verticalLayoutWidget.setObjectName("verticalLayoutWidget")
        self.verticalLayout = QtWidgets.QVBoxLayout(self.verticalLayoutWidget)
        self.verticalLayout.setContentsMargins(0, 0, 0, 0)
        self.verticalLayout.setObjectName("verticalLayout")
        self.label_5 = QtWidgets.QLabel(self.verticalLayoutWidget)
        self.label_5.setStyleSheet("font: 75 10pt \"MS Shell Dlg 2\";")
        self.label_5.setObjectName("label_5")
        self.verticalLayout.addWidget(self.label_5)
        self.p_token = QtWidgets.QLineEdit(self.verticalLayoutWidget)
        self.p_token.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.p_token.setObjectName("p_token")
        self.verticalLayout.addWidget(self.p_token)
        self.label_6 = QtWidgets.QLabel(self.verticalLayoutWidget)
        self.label_6.setStyleSheet("font: 75 10pt \"MS Shell Dlg 2\";")
        self.label_6.setObjectName("label_6")
        self.verticalLayout.addWidget(self.label_6)
        self.s_token = QtWidgets.QLineEdit(self.verticalLayoutWidget)
        self.s_token.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.s_token.setObjectName("s_token")
        self.verticalLayout.addWidget(self.s_token)
        self.connect_button = QtWidgets.QPushButton(self.verticalLayoutWidget)
        self.connect_button.setStyleSheet("font: 75 10pt \"MS Shell Dlg 2\";\n"
"background-color: rgb(184, 184, 184);")
```

```python
"background-color: rgb(184, 184, 184);")
        self.connect_button.setObjectName("connect_button")
        self.verticalLayout.addWidget(self.connect_button)
        self.line = QtWidgets.QFrame(self.tab_user)
        self.line.setGeometry(QtCore.QRect(360, 30, 21, 551))
        self.line.setFrameShape(QtWidgets.QFrame.VLine)
        self.line.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.line.setObjectName("line")
        self.connection_browser = QtWidgets.QTextBrowser(self.tab_user)
        self.connection_browser.setGeometry(QtCore.QRect(460, 140, 311, 281))
        self.connection_browser.setStyleSheet("background-color: rgb(217, 217, 217);")
        self.connection_browser.setObjectName("connection_browser")
        self.tabWidget.addTab(self.tab_user, "")
        self.tab_daily = QtWidgets.QWidget()
        self.tab_daily.setObjectName("tab_daily")
        self.date_lbl = QtWidgets.QLabel(self.tab_daily)
        self.date_lbl.setGeometry(QtCore.QRect(160, 20, 91, 16))
        self.date_lbl.setObjectName("date_lbl")
        self.get_btn = QtWidgets.QPushButton(self.tab_daily)
        self.get_btn.setGeometry(QtCore.QRect(280, 50, 93, 21))
        self.get_btn.setStyleSheet("border-width: 5px;\n"
"border-color: rgb(12, 255, 20);\n"
"background-color: rgb(246, 254, 255);\n"
"")
        self.get_btn.setObjectName("get_btn")
        self.sym_lbl = QtWidgets.QLabel(self.tab_daily)
        self.sym_lbl.setGeometry(QtCore.QRect(20, 20, 81, 16))
        self.sym_lbl.setObjectName("sym_lbl")
        self.sym_in = QtWidgets.QLineEdit(self.tab_daily)
        self.sym_in.setGeometry(QtCore.QRect(20, 50, 113, 22))
        self.sym_in.setStyleSheet("border-width: 5px;\n"
"border-color: rgb(12, 255, 20);\n"
"background-color: rgb(246, 254, 255);\n"
"")
        self.sym_in.setObjectName("sym_in")
        self.date_in = QtWidgets.QDateEdit(self.tab_daily)
        self.date_in.setGeometry(QtCore.QRect(150, 50, 110, 22))
        self.date_in.setStyleSheet("border-width: 5px;\n"
"border-color: rgb(12, 255, 20);\n"
"background-color: rgb(246, 254, 255);\n"
"")
        self.date_in.setObjectName("date_in")
        self.MplGraph = MplGraph(self.tab_daily)
        self.MplGraph.setGeometry(QtCore.QRect(20, 100, 861, 431))
        self.MplGraph.setStyleSheet("background-color: rgb(170, 170, 170);")
        self.MplGraph.setObjectName("MplGraph")
        self.tabWidget.addTab(self.tab_daily, "")
        self.tab_news = QtWidgets.QWidget()
```

GUI - Instance

Inherit QMainWindow and the gui file we just looked at.

Set up connections for actions.

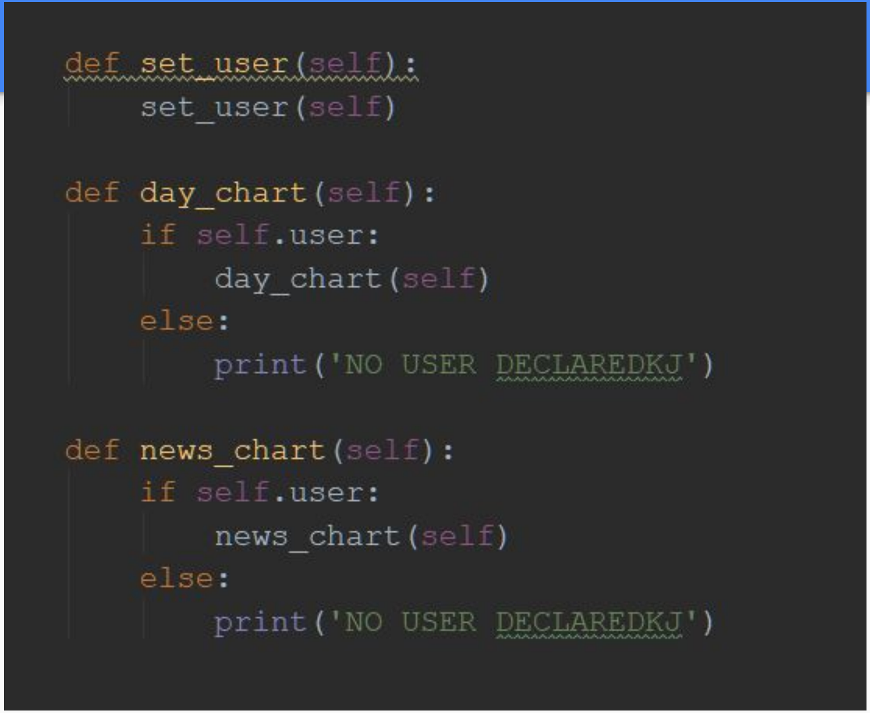


```
class MainGUI(QMainWindow, Ui_Python_IEX_GUI):

    def __init__(self):
        super(MainGUI, self).__init__()

        # USER INFO
        self.user = None
        # set up ui from designer
        self.ui = Ui_Python_IEX_GUI()
        self.ui.setupUi(self)

        # init date
        self.ui.date_in.setDate(datetime.today())

        # set up connections
        self.ui.connect_button.clicked.connect(self.set_user)
        self.ui.get_btn.clicked.connect(self.day_chart)
        self.ui.news_button.clicked.connect(self.news_chart)

        # extras
        self.addToolBar(NavigationToolbar(self.ui.MplGraph.canvas, self.ui.MplGraph)
        self.ui.tabWidget.setCurrentIndex(0)
```

GUI Member Functions

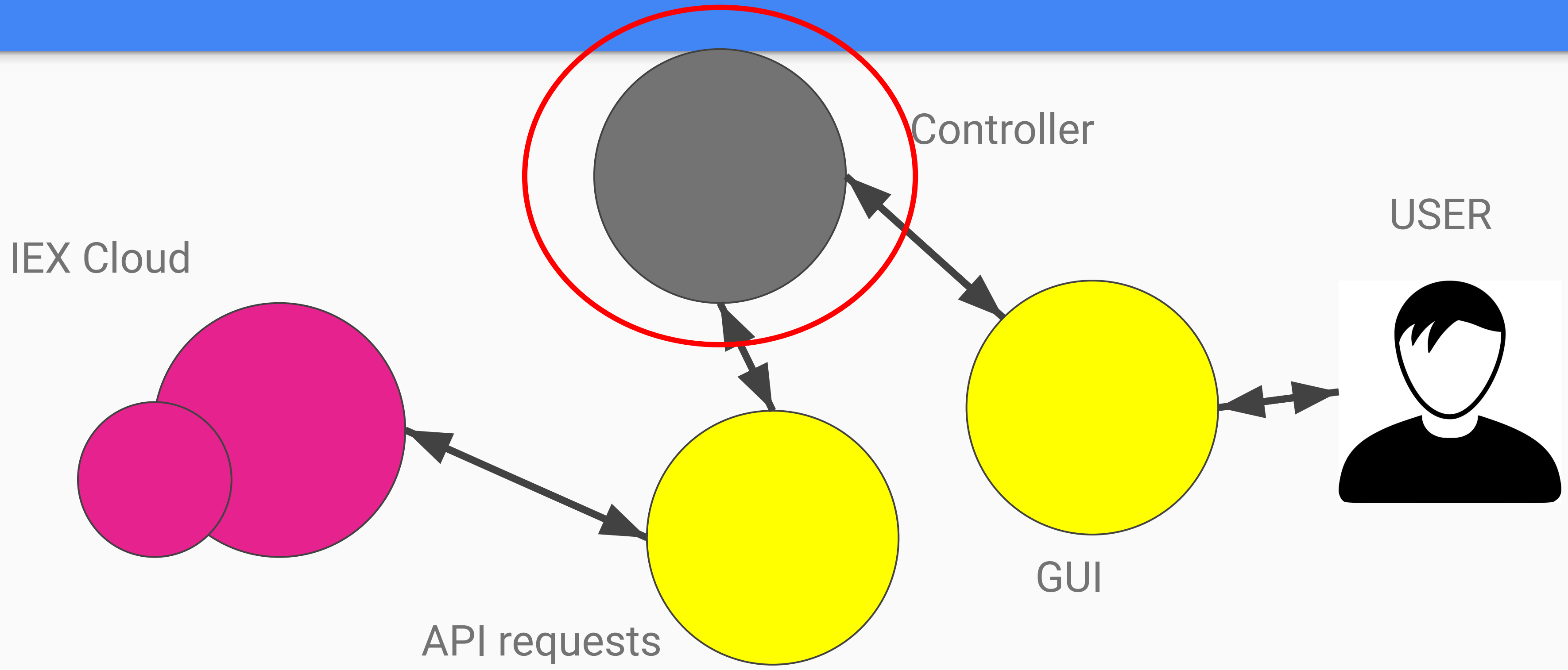
Connections were set to call these functions.

These functions pass the GUI instance itself to other functions that were imported from the "Controller".



```
def set_user(self):
    set_user(self)

def day_chart(self):
    if self.user:
        day_chart(self)
    else:
        print('NO USER DECLAREDKJ')

def news_chart(self):
    if self.user:
        news_chart(self)
    else:
        print('NO USER DECLAREDKJ')
```

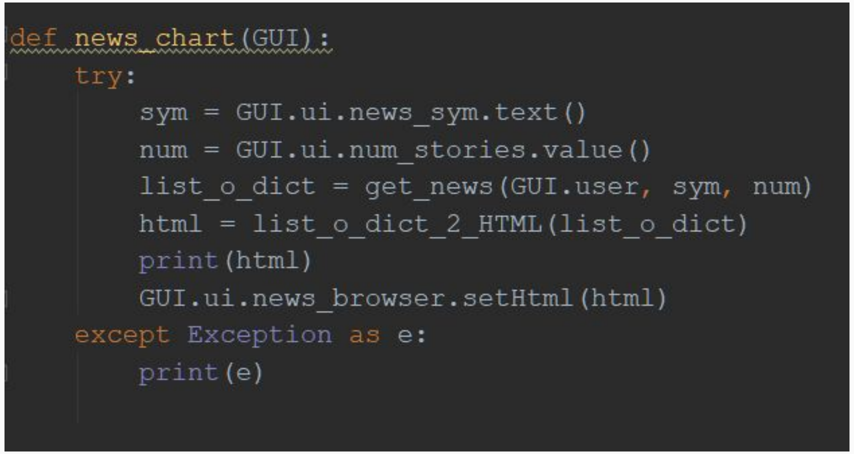
Now let’s look at the “controller”



“Controller”

Manage interaction between GUI and api requests

1. Called in response to GUI event.
2. Pulls attributes from the GUI.
3. Formats calls to api requet scripts.
4. Receives and formats api returns.
5. Injects media back into the GUI.



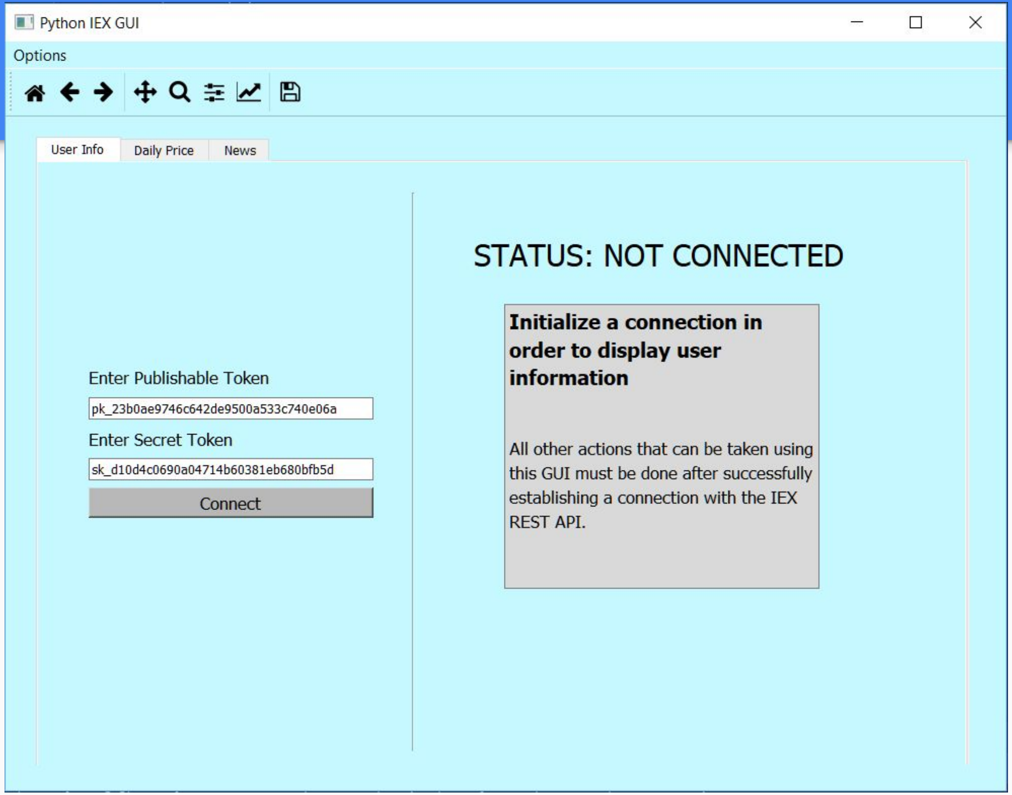
```
def news_chart(GUI):
    try:
        sym = GUI.ui.news_sym.text()
        num = GUI.ui.num_stories.value()
        list_o_dict = get_news(GUI.user, sym, num)
        html = list_o_dict_2_HTML(list_o_dict)
        print(html)
        GUI.ui.news_browser.setHtml(html)
    except Exception as e:
        print(e)
```

DEMO?

Landing Page

Currently initializes with my correct tokens so I don't have to go get them every time.

Pressing “connect” will builds a user instance and check to see if a connection is valid.



Enter Publishable Token

pk_23b0ae9746c642de9500a533c740e06a

Enter Secret Token

sk_d10d4c0690a04714b60381eb680bfb5d

Connect

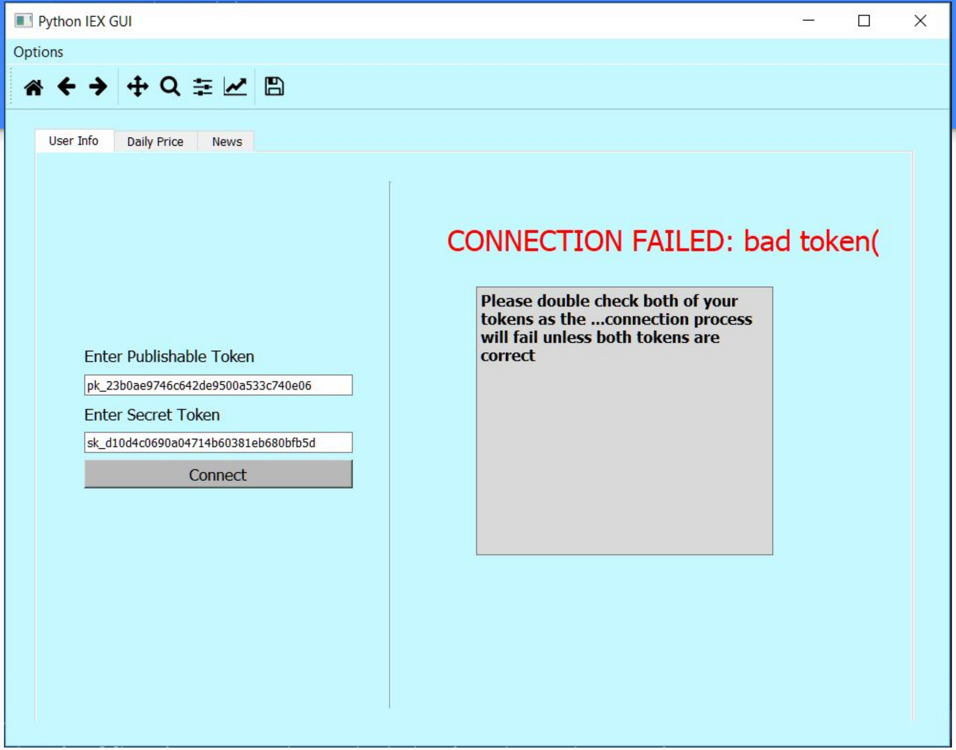
STATUS: NOT CONNECTED

Initialize a connection in order to display user information

All other actions that can be taken using this GUI must be done after successfully establishing a connection with the IEX REST API.

Bad Token

If I delete one letter from the either one of the tokens, the connection will fail and I will get a failure message.



Enter Publishable Token

pk_23b0ae9746c642de9500a533c740e06

Enter Secret Token

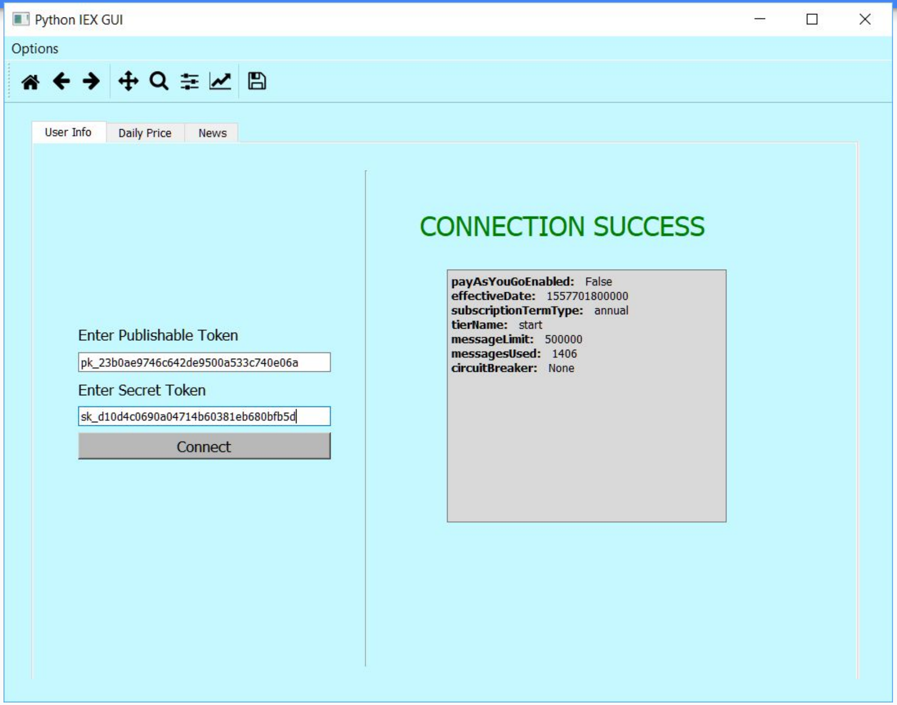
sk_d10d4c0690a04714b60381eb680bfb5d

CONNECTION FAILED: bad token(

Please double check both of your tokens as the ...connection process will fail unless both tokens are correct

Good Token

If both tokens are correct you will get a “CONNECTION SUCCESS” notice along with a print out about your account and usage information.



Enter Publishable Token

pk_23b0ae9746c642de9500a533c740e06a

Enter Secret Token

sk_d10d4c0690a04714b60381eb680bfb5d

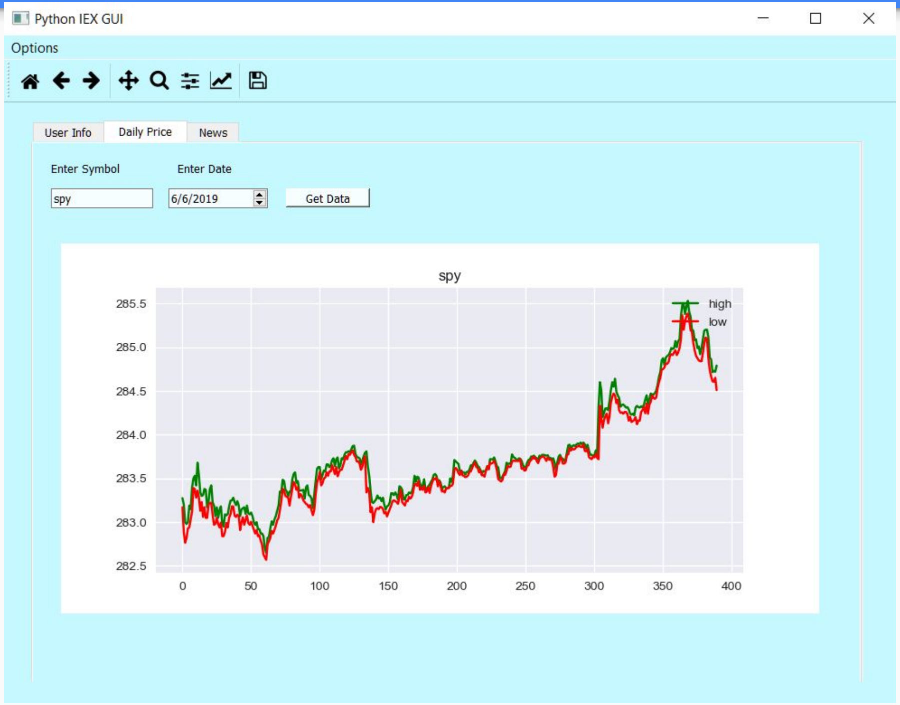
CONNECTION SUCCESS

```
payAsYouGoEnabled:   False
effectiveDate:   1557701800000
subscriptionTermType:   annual
tierName:   start
messageLimit:   500000
messagesUsed:   1406
circuitBreaker:   None
```

Daily Ticker

Enter a symbol, get the daily ticker for that symbol… simple as that.

Date functions are currently not working due to misunderstanding of API implementation.

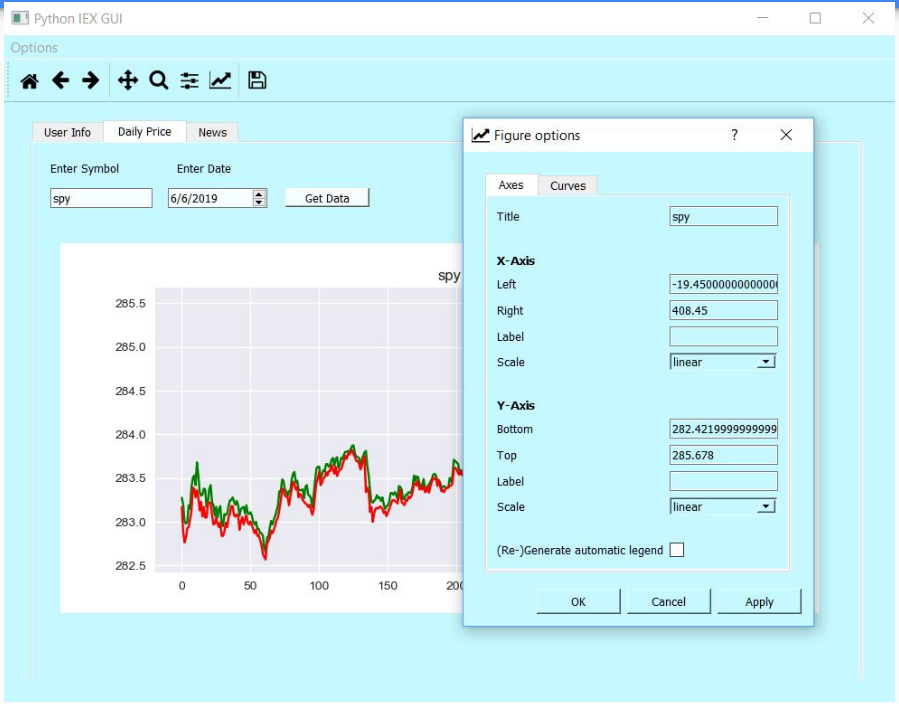


Enter Symbol: spy

Enter Date: 6/6/2019

Toolbar

Since the Matplotlib toolbar was added to the project we have all the functionality of that toolbar when dealing with Matplotlib graphs.



Enter Symbol: spy

Enter Date: 6/6/2019

Title: spy

X-Axis

Left: -19.4500000000000

Right: 408.45

Label:

Scale: linear

Y-Axis

Bottom: 282.4219999999999

Top: 285.678

Label:

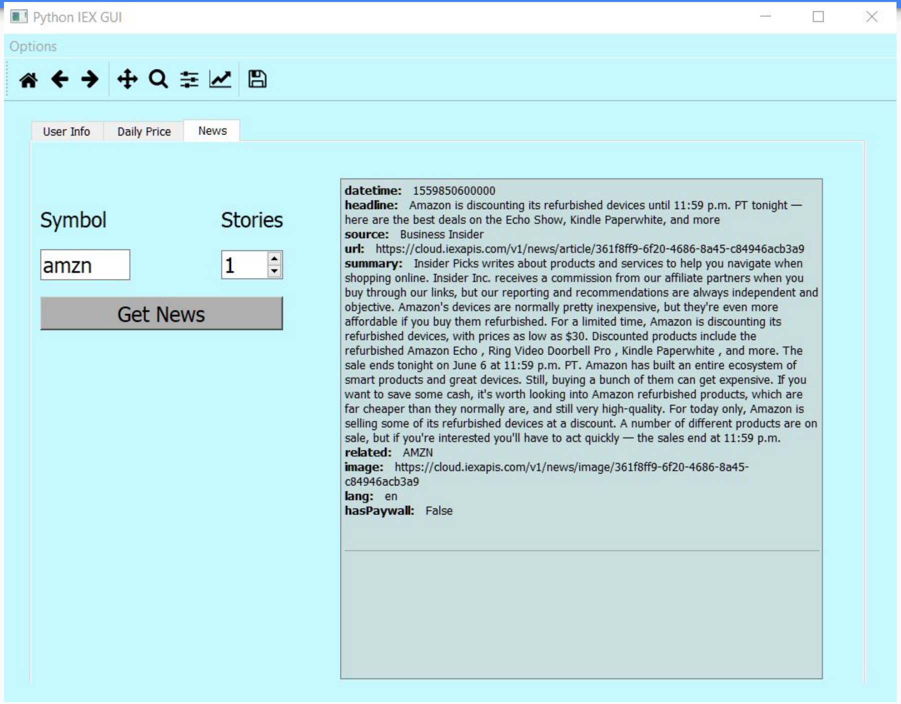
Scale: linear

(Re-)Generate automatic legend ☐

Get a news clip

Since we are at the CSLU I thought it would be interesting to have access to live news feed data from my tool.

You enter a stock symbol and a news clip related to the stock will be returned.



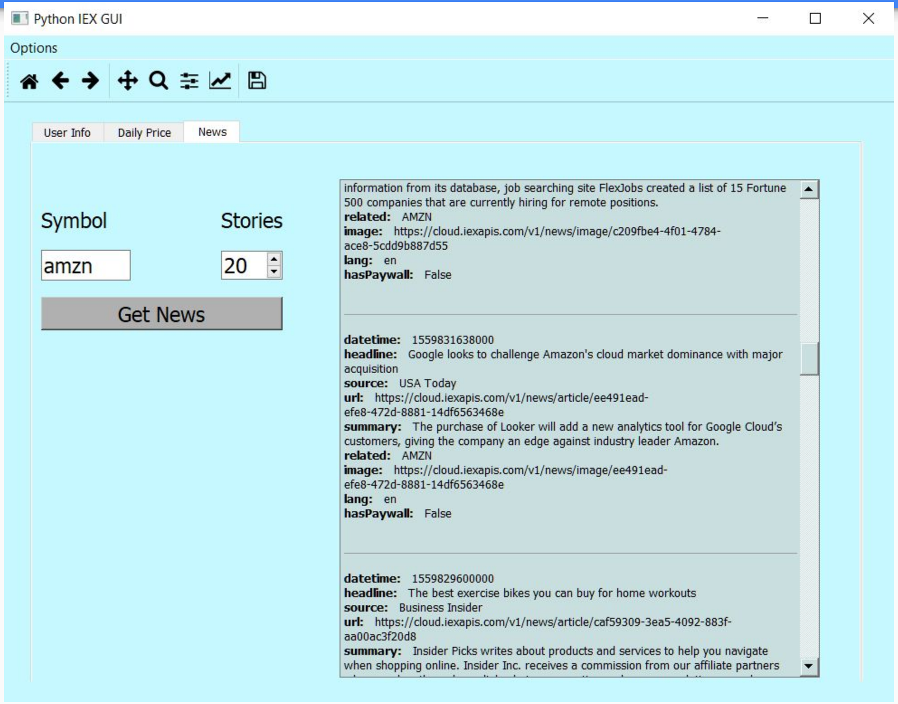
Symbol: amzn

Stories: 1

datetime: 1559850600000
headline: Amazon is discounting its refurbished devices until 11:59 p.m. PT tonight — here are the best deals on the Echo Show, Kindle Paperwhite, and more
source: Business Insider
url: https://cloud.iexapis.com/v1/news/article/361f8ff9-6f20-4686-8a45-c84946acb3a9
summary: Insider Picks writes about products and services to help you navigate when shopping online. Insider Inc. receives a commission from our affiliate partners when you buy through our links, but our reporting and recommendations are always independent and objective. Amazon's devices are normally pretty inexpensive, but they're even more affordable if you buy them refurbished. For a limited time, Amazon is discounting its refurbished devices, with prices as low as $30. Discounted products include the refurbished Amazon Echo , Ring Video Doorbell Pro , Kindle Paperwhite , and more. The sale ends tonight on June 6 at 11:59 p.m. PT. Amazon has built an entire ecosystem of smart products and great devices. Still, buying a bunch of them can get expensive. If you want to save some cash, it's worth looking into Amazon refurbished products, which are far cheaper than they normally are, and still very high-quality. For today only, Amazon is selling some of its refurbished devices at a discount. A number of different products are on sale, but if you're interested you'll have to act quickly — the sales end at 11:59 p.m.
related: AMZN
image: https://cloud.iexapis.com/v1/news/image/361f8ff9-6f20-4686-8a45-c84946acb3a9
lang: en
hasPaywall: False

Get many news clips

This functionality supports up to 50 news clips returned at a time.



```
information from its database, job searching site FlexJobs created a list of 15 Fortune
500 companies that are currently hiring for remote positions.
related:   AMZN
image:   https://cloud.iexapis.com/v1/news/image/c209fbe4-4f01-4784-
ace8-5cdd9b887d55
lang:   en
hasPaywall:   False

datetime:   1559831638000
headline:   Google looks to challenge Amazon's cloud market dominance with major
acquisition
source:   USA Today
url:   https://cloud.iexapis.com/v1/news/article/ee491ead-
efe8-472d-8881-14df6563468e
summary:   The purchase of Looker will add a new analytics tool for Google Cloud's
customers, giving the company an edge against industry leader Amazon.
related:   AMZN
image:   https://cloud.iexapis.com/v1/news/image/ee491ead-
efe8-472d-8881-14df6563468e
lang:   en
hasPaywall:   False

datetime:   1559829600000
headline:   The best exercise bikes you can buy for home workouts
source:   Business Insider
url:   https://cloud.iexapis.com/v1/news/article/caf59309-3ea5-4092-883f-
aa00ac3f20d8
summary:   Insider Picks writes about products and services to help you navigate
when shopping online. Insider Inc. receives a commission from our affiliate partners
```

Major difficulties

Getting matplotlib to work with PyQt. Many extra steps involving specialized backend packages within matplotlib.

Mobil API: IEX is constantly updating its api and some features that were available when I started this project are not the same now.

‘\0’

Thank you!