FinMesh

Release 2021

Michael Hartmann

Aug 24, 2022

TABLE OF CONTENTS:

FinMesh is a python-based package that brings together financial data from various sources in one place for ease of use and distribution. The four main sections of FinMesh are (1) the IEX REST API, (2) data from the US treasury, data from the US Federal Reserve Economic Data, and (4) data from the SECs EDGAR system.

The purpose of this package and its sub-packages was originally to practice interacting with API data. With third-party API packages there is always the risk of outages or bugs. In building the original IEX wrapper we sought to build something easy to understand and use, that can be updated quickly and accurately.

With the addition of the US Federal data the opportunity arose to create a package that could deliver all sorts of economic and security data from one place. In doing so we hope to create a low-barrier way for beginners to play with large and very useful data sets.

In the future, this package will be updated with new financial and economic APIs. If you know of a low or no cost API that could be incorporated please raise it as an issue and we will work to have it done ASAP.

You can donate to this project HERE. If FinMesh has brought you value please consider supporting the project.

CHAPTER

ONE

INSTALLATION

Install FinMesh using pip for a tested, up-to-date version:

```
pip install FinMesh
```

1.1 Dependencies and Module-Specific Requirements

Depending on your use-case, there may be dependencies that are required for complete functionality. If you are only interested in accessing IEX Cloud, `pandas` will be the only required install. Below is a list of non-standard packages that are required and the modules that require them:

- Pandas [iex] `pip install pandas`
- xmltodict [usgov] `pip install xmltodict`
- Beautiful Soup [edgar] `pip install bs4`
- Natural Language Toolkit [edgar] `pip install nltk`

1.2 API Token Setup

Some APIs require authentication through the use of tokens. These tokens should be set up as environment variables in the bash profile. A great article on how to do this on Mac is available here:

My Mac OSX Bash Profile

Or you can follow the guide down below for Mac and Linux.

Click HERE for your free IEX token. This token must be stored as IEX_TOKEN in your environment variables.

Click HERE for your free FRED token. This token must be stored as FRED_TOKEN in your environment variables.

1.2.1 What is an Environment Variable?

Environment variables are used in FinMesh to store API tokens and token-related data. Tokens are like passwords that API providers use to authenticate the requests being made. This ensures that (1) unauthorized parties cannot access the data or spam the API with bogus requests (such as in DDOS attacks) and (2) users can track their useage and manage permissions. The IEX Cloud and FRED APIs both use tokens to authenticate users, and IEX in particular uses tokens to track useage and manage the level of services available.

In these examples we will be setting global environment variables using the standard bash profiles available on both MacOS and Linux. You'll want to fire up a terminal window now so you can follow along.

For Mac users, your bash profile is called `.bash_profile`, pretty simple. When you open this, you'll likely find an empty file.

For Linux users, your bash profile is called `.bashrc`. Depending on your distribution, you may or may not find some pre-made bash shortcuts and other assorted code inside, we don't need to worry about any of that right now.

Your bash profile is stored in the User's home directory, this is the directory that a new terminal window will automatically open into so if you have just opened a new window, you're in the right place! This directory is most commonly denoted with a '~' before a '$'. If you have navigated into a folder, simply type 'cd' to return to the home directory. Now that you are in the correct directory lets open your bash profile using nano. Nano is simply a text editor that is built into terminal. Type this and then hit enter, and you'll be taken to your bash profile.

Mac:

```
nano .bash_profile
```

Linux:

```
nano .bashrc
```

1.2.2 Setting Up Your Token Variables

Now that you are ready to add some token variables, you can go ahead and collect them from IEX and/or FRED from the links at the beginning of this page.

In the standard `IEX_TOKEN` variable, you'll want to paste your PUBLIC token from IEX. This way if you accidentally publish this code, or otherwise have the token compromised you can simply cut that token off from API privileges. You can of course use the secret key, but you are setting yourself up for a security vulnerability.

The `IEX_SANDBOX_TOKEN` is also fairly self-explanatory. You can find this by switching to 'Sandbox Mode' in the IEX Console and looking where your regular tokens would normally be.

The `SANDBOX` boolean variable will tell FinMesh whether you want to access Sandbox data instead of the real data. This is really useful when you are testing code and don't want to use credits.

The `FRED` token is even more straight forward. There are no sandbox options and no public/private versions of keys. Simply paste in your token and you are good to go!

Once you've written out the code below and filled in the blanks with the appropriate tokens, your bash profile is all set:

```
# FinMesh Token Setup

# Public IEX Cloud Token
IEX_TOKEN=#PLACE-YOUR-TOKEN-HERE#
# Sandbox IEX Token
IEX_SANDBOX_TOKEN=#PLACE-YOUR-TOKEN-HERE#
```

(continues on next page)

(continued from previous page)

```
# Sandbox State
SANBOX=False
# FRED Token
FRED_TOKEN=#PLACE-YOUR-TOKEN-HERE#
```

You can now exit nano by hitting `CTRL+O` to writeout(save) your work and then `CTRL+X` to close the nano editor. Once you are back into the terminal, type the following to apply the changes to your environment.

Mac:

```
source .bash_profile
```

Linux:

```
source .bashrc
```

1.2.3 Conclusion

You are now all set up to use FinMesh! If you have questions or concerns you are always free to reach out to me at the links below! In the future this type of setup may not be neccesary as we look at other ways to supply tokens that retains the security and seamlessness of this method while being easier to configure.

CHAPTER

TWO

ACCESSING IEX CLOUD

These tutorials are meant for those who are new to programming, and cover each step fairly thoroughly. If you are experienced in Python, the docs are more in depth and cover the actual methods available.

2.1 Using the IEXStock Class to Access Data

The easiest way to access data for a particular stock (or group of stocks) is using the IEXStock class. It stores relevant information like ticker, date, and data period preferences so you can request data and get it in a single line. It also stores all the requests you've made as dynamic class attributes, meaning you can save on credits. You can output data as JSON or as Pandas, and then through Pandas as either excel or csv format.

The IEXStock class is contained in the `iex` directory's `__init__` file so there is no need to specify further than the directory level.

2.1.1 Importing and Initializing

When you attach the variable name `AAPL` to the class, you are doing what is called initializing the class. When you initialize a class, sometimes you are required to input some details about the class. These are called arguments. The IEXStock class only requires a single argument, and that is `'ticker'`. This is the ticker or symbol of the stock as a string.

The program will tell you if you forgot an argument and that will look something like this:

```
>>> AAPL = IEXStock()
TypeError: __init__() missing 1 required positional argument: 'ticker'
```

In these examples I'll look at three ways one can get FinMesh into their code. They all accomplish the same thing.

You can import the whole package and call the directory and class:

```
>>> import FinMesh
>>> AAPL = FinMesh.iex.IEXStock('AAPL')
```

You can import the specific IEX module and call the class:

```
>>> from FinMesh import iex
>>> AAPL = iex.IEXStock('AAPL')
```

Or you can import just the class and save yourself some typing:

```
>>> from FinMesh.iex import IEXStock
>>> AAPL = IEXStock('AAPL')
```

2.1.2 Calling Methods - The Basics

Once you have initialized a new class with the ticker of the stock you want to look at, you can start requesting data. Each grouping of data (in API speak we would call them endpoints) uses a function to gather and optionally format the data. The naming of all the methods in the IEXStock class follow two rigid rules:

1. All method names will match as closely as possible with their names according to IEX Cloud documentation, with all spaces as underscores

and

2. All data request methods under the IEXStock class have the prefix 'get_'

Here we will request some key stats about the company and their latest quarterly balance sheet.

```
>>> AAPL.get_key_stats()
# A whole bunch of JSON containing the keys and values from the Key Stats endpoint
```

```
>>> AAPL.get_balance_sheet()
# A whole bunch of JSON containing the keys and values from the latest balance sheet
```

These methods will return some nice JSON data, but we don't always want JSON data. Pandas is the solution for that. Pandas takes data and formats it into `dataframes`, otherwise known as tables. This makes it easier to work with in Python and export to Excel. If you want to output a Pandas dataframe, simply specify the output:

```
>>> AAPL.get_key_stats(ouput='pandas')
# A nice dataframe containing the keys and values from the Key Stats endpoint
```

2.1.3 Class Attributes

Every time you call a method, you are making a request to the IEX Cloud API, thus costing you credits. Even the lowest teir has more than enough credits to work with small projects, but as soon as you want to start gathering data on a larger scale, you will find yourself needing more credits..

One limited solution to this is keeping the responses from the methods as a class attribute. A class attribute is basically just a characteristic of the class. For example, an attribute of moles is they live undergound, and an attribute of our example IEXStock class is the key stats data.

When you call a method, the result will automatically be assigned to an attribute of the same name, minus the 'get' prefix. You call it with the class name and no parenthesis.

```
>>> AAPL.key_stats
# A nice dataframe containing the keys and values from the Key Stats endpoint
```

```
>>> AAPL.balance_sheet
# A whole bunch of JSON containing the keys and values from the latest balance sheet
```

Notice that when we call the attributes they return different data types. That is because the attribute will be whatever output type (JSON or DataFrame) was requested last.

2.1.4 Saving and Loading Class Instances

FinMesh uses `Pickle` to save and load states. It's a Python standard library and is perfect for the task. When we 'pickle' something (like a cucumber or a class instance) we are preserving it for later.

Pickling is useful because every time you call a class method, you are using IEX Cloud credits. To keep from having to make the same request over and over, we can save the class (and the information that has been assigned to a class attribute) for later. Pickling is fairly straight forward. Let's say we create an instance of IEXStock for AAPL, and call the key stats method. Then we will pickle the result to save it for later:

```
>>> AAPL = IEXStock('AAPL')
>>> AAPL.get_key_stats()
# A whole bunch of JSON containing the keys and values from the Key Stats endpoint
>>> AAPL.pickle_class_state()
# Output a file with the pickled class inside.
```

In this example, assuming the date is January 27th, 2022, the output file would be called `'AAPL_2022-01-27.pickle'`. An internal method takes care of naming the file so that every one is predictable and standard.

In order to load a pickled class, we can call the `unpickle_class_state(*file*)` method, specifying the file name of the pickled state.

2.2 Using Base methods to Access Data

The IEXStock class is built on a collection of sub-modules containing simple methods that request and receive JSON. These have no Pandas output and are purposefully minimum viable methods for accessing IEX Cloud data.

The base methods for IEX Cloud are contained in the `stock`, `premium`, `market`, and `forex` modules. You will need to specify which module you would like to access in the `import` path. All the methods you need for stock specific data will be in the `stock` sub-module.

CHAPTER

THREE

IEX CLOUD MODULES

3.1 Stock Endpoint

`iex.stock.balance_sheet(symbol, external=False, vprint=False, **query_params)`

Returns
Balance sheet financial statement for the requested stock.

Parameters

- **symbol** (*string, required*) – The ticker or symbol of the stock you would like to request.
- **queries** (*key value pair where key is variable and value is string*) – Standard kwargs parameter.

`iex.stock.book(symbol, external=False, vprint=False)`

Returns
Book price for the requested stock.

Parameters
symbol (*string, required*) – The ticker or symbol of the stock you would like to request.

`iex.stock.cash_flow(symbol, external=False, vprint=False, **query_params)`

Returns
Cash sheet financial statment for the requested stock.

Parameters

- **symbol** (*string, required*) – The ticker or symbol of the stock you would like to request.
- **queries** (*key value pair where key is variable and value is string*) – Standard kwargs parameter.

`iex.stock.collection(collectionType, collectionName, external=False, vprint=False)`

Returns
Quotes for stock in the requested collection type.

Parameters

- **collectionType** (*accepted values are ['sector', 'tag', 'list'], required*) – The type of data returned by the endpoint.
- **collectionName** (*string, required*) – Name of the sector, tag, or list to return. List of names available on IEX Cloud.

`iex.stock.company(symbol, external=False, vprint=False, **query_params)`

Returns
Company data such as website, address, and description for the requested company.

Parameters
symbol (*string, required*) – The ticker or symbol of the stock you would like to request.

`iex.stock.delayed_quote(symbol, external=False, vprint=False)`

Returns
15-minute delayed market quote for the requested ticker.

Parameters
symbol (*string, required*) – The ticker or symbol of the stock you would like to request.

`iex.stock.dividends(symbol, scope, external=False, vprint=False)`

Returns
Returns dividend information for a requested ticker.

Parameters

- **symbol** (*string, required*) – The ticker or symbol of the stock you would like to request.
- **scope** (*accepted arguments: ['5y','2y','1y','ytd','6m','3m','1m','next'], required*) – The range of data needed.

`iex.stock.earnings(symbol, last=None, field=None, external=False, vprint=False)`

Returns
Earnings data such as actual EPS, beat/miss, and date for the requested ticker.

Parameters

- **symbol** (*string, required*) – The ticker or symbol of the stock you would like to request.
- **last** (*string, optional*) – The number of previous earnings to return.
- **field** (*string, optional*) – The specific field from the earnings report ot return.

`iex.stock.estimates(symbol, external=False, vprint=False)`

Returns
Latest future earnings estimates for the requested ticker.

Parameters
symbol (*string, required*) – The ticker or symbol of the stock you would like to request.

`iex.stock.financials(symbol, period=None, external=False, vprint=False)`

Returns
Brief overview of a company's financial statements.

Parameters

- **symbol** (*string, required*) – The ticker or symbol of the stock you would like to request.
- **period** (*accepted values are ['annual', 'quarterly'], optional*) – The time interval of financial statements returned.

`iex.stock.fund_ownership(symbol, external=False, vprint=False)`

Returns
Largest 10 fund owners of the requested ticker. This excludes explicit buy or sell-side firms.

Parameters
symbol (*string, required*) – The ticker or symbol of the stock you would like to request.

`iex.stock.historical_price(symbol, period, date=None, external=False, vprint=False, **query_params)`

Returns
Adjusted and unadjusted historical data for up to 15 years, and historical minute-by-minute intraday prices for the last 30 trailing calendar days.

Parameters

- **symbol** (*string, required*) – The ticker or symbol of the stock you would like to request.
- **period** (*string, required*) – The period of data you would like to have returned. Accepted arguments are ['max', '5y', '2y', '1y', 'ytd', '6m', '3m', '1m', '1mm', '5d', '5dm', 'date', 'dynamic']

`iex.stock.income_statement(symbol, external=False, vprint=False, **query_params)`

Returns
Income statement financial data for the requested ticker.

Parameters

- **symbol** (*string, required*) – The ticker or symbol of the stock you would like to request.
- **queries** (*key value pair where key is variable and value is string*) – Standard kwargs parameter.

`iex.stock.insider_roster(symbol, external=False, vprint=False)`

Returns
10 largest insider owners for the requested ticker.

Parameters
symbol (*string, required*) – The ticker or symbol of the stock you would like to request.

`iex.stock.insider_summary(symbol, external=False, vprint=False)`

Returns
Summary of the insiders and their actions within the last 6 months for the requested ticker.

Parameters
symbol (*string, required*) – The ticker or symbol of the stock you would like to request.

`iex.stock.insider_transactions(symbol, external=False, vprint=False)`

Returns
Summary of insider transactions for the requested ticker.

Parameters
symbol (*string, required*) – The ticker or symbol of the stock you would like to request.

`iex.stock.institutional_ownership(symbol, external=False, vprint=False)`

Returns
10 largest instituional owners for the requested ticker. This is defined as explicitly buy or sell-side only.

Parameters
symbol (*string, required*) – The ticker or symbol of the stock you would like to request.

`iex.stock.ipo_today(external=False, vprint=False)`

Returns
List of IPOs happening today.

`iex.stock.ipo_upcoming(external=False, vprint=False)`

Returns
List of upcoming IPOs for the current and next month.

`iex.stock.key_stats(symbol, stat=False, external=False, vprint=False)`

Returns
Important and key statistics for the requested ticker.

Parameters

- **symbol** (*string, required*) – The ticker or symbol of the stock you would like to request.
- **stat** (*string, optional*) – The specific stat which you would like to return.

`iex.stock.largest_trades(symbol, external=False, vprint=False)`

Returns
Delayed list of largest trades for the requested ticker.

Parameters
symbol (*string, required*) – The ticker or symbol of the stock you would like to request.

`iex.stock.logo(symbol, external=False, vprint=False)`

Returns
Google APIs link to the logo for the requested ticker.

Parameters
symbol (*string, required*) – The ticker or symbol of the stock you would like to request.

`iex.stock.market_list(list_type, display_percent=None, external=False, vprint=False)`

Returns
10 largest companies in the specified list.

Parameters

- **list_type** (*accepted values are ['mostactive', 'gainers', 'losers', 'iexvolume', 'iexpercent', 'premarket_losers', 'postmarket_losers', 'premarket_gainers', 'postmarket_gainers'], required*) – The list that you would like to return.
- **displayPercent** (*boolean, optional*) – Whether you would like to see the percentage values multiplied by 100

`iex.stock.market_volume(format=None, external=False, vprint=False)`

Returns
Market wide trading volume.

Parameters
format (*accepted value is 'csv', optional*) – The output format of the endpoint

`iex.stock.new_historical_price(symbol, period, date=None, chartByDay=False, external=False, vprint=False, **query_string_params)`

Returns
Adjusted and unadjusted historical data for up to 15 years, and historical minute-by-minute intraday prices for the last 30 trailing calendar days.

Parameters

- **symbol** (*string, required*) – The ticker or symbol of the stock you would like to request.
- **period** (*string, required*) – The period of data you would like to have returned. Accepted arguments are ['max', '5y', '2y', '1y', 'ytd', '6m', '3m', '1m', '1mm', '5d', '5dm', 'date', 'dynamic']
- **date** (*string, optional*) – If used with the query parameter chartByDay, then this returns historical OHLCV data for that date. Otherwise, it returns data by minute for a specified date. Date format YYYYMMDD
- **chartByDay** (*boolean, optional*) – If single date is specified, this returns historical OHLCV data for that date.

Query string parameters allow you to specify what data you want on a finer scale. Boolean parameters should be typed as strings in the following format: '`key=value`'

A full list of these parameters can be found in the IEX documentation.

`iex.stock.news`(*symbol*, *last=None*, *external=False*, *vprint=False*)

Returns
News item summaries for the requested ticker.

Parameters

- **symbol** (*string, required*) – The ticker or symbol of the stock you would like to request.
- **last** (*integer, optional*) – The number of news items to return.

`iex.stock.ohlc`(*symbol*, *external=False*, *vprint=False*)

Returns
Most recent days open, high, low, and close data for the requested ticker.

Parameters
symbol (*string, required*) – The ticker or symbol of the stock you would like to request.

`iex.stock.peers`(*symbol*, *external=False*, *vprint=False*)

Returns
List of a requested ticker's peers.

Parameters
symbol (*string, required*) – The ticker or symbol of the stock you would like to request.

`iex.stock.previous`(*symbol*, *external=False*, *vprint=False*)

Returns
Previous day's price data for the requested ticker.

Parameters
symbol (*string, required*) – The ticker or symbol of the stock you would like to request.

`iex.stock.price`(*symbol*, *external=False*, *vprint=False*)

Returns
Single float value of the requested ticker's price.

Parameters
symbol (*string, required*) – The ticker or symbol of the stock you would like to request.

iex.stock.**price_target**(*symbol*, *external=False*, *vprint=False*)

Returns
Analyst's price targets for the requested ticker.

Parameters
symbol (*string, required*) – The ticker or symbol of the stock you would like to request.

iex.stock.**quote**(*symbol*, *field=None*, *external=False*, *vprint=False*)

Returns
rice quote data for the requested ticker. Fields are able to be called individually.

Parameters

- **symbol** (*string, required*) – The ticker or symbol of the stock you would like to request.
- **field** (*string, optional*) – The specific field from the quote endpoint you would like to return.

iex.stock.**recommendation_trends**(*symbol*, *external=False*, *vprint=False*)

Returns
Analyst recommendations for the requested ticker.

Parameters
symbol (*string, required*) – The ticker or symbol of the stock you would like to request.

iex.stock.**sector_performance**(*external=False*, *vprint=False*)

Returns
Market performance for all sectors.

iex.stock.**splits**(*symbol*, *scope=None*, *external=False*, *vprint=False*)

Returns
Record of stock splits for the requested ticker.

Parameters

- **symbol** (*string, required*) – The ticker or symbol of the stock you would like to request.
- **scope** (*accepted arguments: ['5y','2y','1y','ytd','6m','3m','1m','next'], optional*) – The range of data needed.

iex.stock.**today_earnings**(*external=False*, *vprint=False*)

Returns
Earnings data released today, grouped by timing and stock.

iex.stock.**volume_by_venue**(*symbol*, *external=False*, *vprint=False*)

Returns
rading volume for the requested ticker by venue.

Parameters
symbol (*string, required*) – The ticker or symbol of the stock you would like to request.

3.2 Market Endpoint

`iex.market.commodities(symbol, vprint=False)`

Returns
Commodites data for the requested commodities symbol.

Parameters
symbol (*string, required*) – The symbol of the commodity you would like to return.

`iex.market.economic_data(symbol, vprint=False)`

Returns
Economic data for the requested economic indicator symbol.

Parameters
symbol (*string, required*) – The symbol of the economic indicator you would like to return.

`iex.market.generic_data_point(symbol, key=None, vprint=False)`

Returns
Generic endpoint used to access all 'Data Point' data sets on IEX

Parameters
symbol (*string, required*) – The symbol of the data point you would like to return.

`iex.market.generic_time_series(symbol, *args, vprint=False)`

Returns
Generic endpoint used to access all 'Time Series' data sets on IEX

Parameters
symbol (*string, required*) – The symbol of the time series data you would like to return.

3.3 Forex Endpoint

`iex.forex.forex_conversion(symbols, amount, vprint=False)`

Returns
Converts one currency to another using up-to-date currency information.

Parameters

- **symbol** (*string, required*) – The symbol of the currency pair you would like to return.
- **amount** (*float, required*) – The amount of the primary currency you wish to calculate and exchange rate on.

`iex.forex.forex_historical(symbols, vprint=False, **queries)`

Returns
Historical FOREX rates for the requested currency pair symbol.

Parameters
symbol (*string, required*) – The symbol of the currency pair you would like to return.

`iex.forex.forex_latest_rate(symbols, vprint=False)`

Returns
Latest FOREX rate for the requested currency pair symbol.

Parameters
symbol (*string, required*) – The symbol of the currency pair you would like to return.

CHAPTER

FOUR

IEXSTOCK CLASSES

4.1 IEXStock Class

class iex.**IEXStock**(*ticker*, *period='quarter'*, *last=1*, *autopopulate=False*)

A class that is built around retrieving data from the IEX Cloud API service. All available data is derived from functions defined in the stock.py module, and are implemented here with a *get_* prefix. All data retrieved is automatically stored in the corrosponding class attribute (sans *get_* prefix). Data set to class attributes can be saved to file and subsequently loaded from that file to limit credit usage in IEX Cloud.

Parameters

- **ticker** (*string, required*) – The symbol or ticker of the stock for which the class is to be created.
- **period** (*string, optional*) – Default is 'quarter'. Allows a user to set period data for all income statement requests.
- **last** (*integer, optional*) – Default is 1. Allows a user to specify how many statements to retrieve of financial statement data.
- **autopopulate** (*boolean, optional*) – Automatically populates key information as class attributes. Uses methods 'basic_information' and 'price_information'

basic_information()

6 credits per symbol requested. Makes requests to the company and key stat IEX endpoints. Sets class attributes for:

Company name (self.company_name)

Industry (self.industry)

Market Capitalization (self.market_cap)

P/E Ratio (self.pe_ratio)

Beta (self.beta)

get_advanced_fundementals(*period*, *output=None*)

75,000 credits per symbol requested. CSV is formatted vertically with keys in the first column. Sets class attribute 'advanced_fundementals'

Returns

Immediate access to the data points in IEX models for 2850+ companies. Models are updated daily.

Parameters

- **period** (*accepted values are ['annual', 'quarterly'], optional*) – Time interval of statements. Defaults to quarterly
- **last** (*integer, optional*) – Number of periods to return, up to 4 for annual and 16 for quarterly. Defaults to 1.
- **output** (*accepted values are ['dataframe', 'csv', 'excel'], optional*) – Determines the output of the data. Default is raw JSON.

get_advanced_stats(*output=None*)

3,005 credits per symbol requested. Sets class attribute 'advanced_stats'.

Returns
Buffed version of key stats with selected financial data and more. Includes all data points from 'key stats'.

Parameters
output (*accepted values are ['dataframe', 'csv', 'excel'], optional*) – Determines the output of the data. Default is raw JSON.

get_analyst_recommendations(*output=None*)

Premium Data. 1,000 premium credits per symbol requested. Sets class attribute 'analyst_recommendations'.

Returns
Analyst stock recommendations for the requested stock from the last four months.

Parameters
output (*accepted values are ['dataframe', 'csv', 'excel'], optional*) – Determines the output of the data. Default is raw JSON.

get_balance_sheet(*period=None*, *last=None*, *output=None*)

3,000 credits per symbol requested.

Sets class attribute 'self.balance_sheet'

Returns
Balance sheet data for the requested company. If any non-JSON output is chosen, the method will return a Pandas DataFrame with the data from this endpoint.

Parameters

- **period** (*accepted values are ['annual', 'quarterly'], optional*) – Time interval of statements. Defaults to quarterly
- **last** (*integer, optional*) – Number of periods to return, up to 4 for annual and 16 for quarterly. Defaults to 1.
- **output** (*accepted values are ['dataframe', 'csv', 'excel'], optional*) – Determines the output of the data. Default is raw JSON.

get_basic_financials(*output=None*)

5000 credits per symbol requested. Note that fetching all three full financial statements has the same credit cost as this endpoint. Sets class attribute 'basic_financials'.

Returns
Basic financial data from the requested company.

Parameters
output (*accepted values are ['dataframe', 'csv', 'excel'], optional*) – Determines the output of the data. Default is raw JSON.

get_cash_flow_statement(*period=None*, *last=None*, *output=None*)

1,000 credits per symbol requested.

Sets class attribute 'self.cash_flow_statement'

Returns

Cash flow statement data for the requested company. If any non-JSON output is chosen, the method will return a Pandas DataFrame with the data from this endpoint.

Parameters

- **period** (*accepted values are ['annual', 'quarterly'], optional*) – Time interval of statements. Defaults to quarterly
- **last** (*integer, optional*) – Number of periods to return, up to 4 for annual and 16 for quarterly. Defaults to 1.
- **output** (*accepted values are ['dataframe', 'csv', 'excel'], optional*) – Determines the output of the data. Default is raw JSON.

get_company(*output=None*)

1 credit per symbol requested. Sets class attribute 'company'.

Returns

General information on the company requested.

Parameters

output (*accepted values are ['dataframe', 'csv', 'excel'], optional*) – Determines the output of the data. Default is raw JSON.

get_delayed_quote(*output=None*)

1 credit per symbol requested. Sets class attribute 'delayed_quote'.

Returns

15 minute delayed quote for the requested symbol.

Parameters

output (*accepted values are ['dataframe', 'csv', 'excel'], optional*) – Determines the output of the data. Default is raw JSON.

get_dividends(*scope*, *output=None*)

10 credits per symbol requested. Sets class attribute 'dividends'.

Returns

Basic dividend information for the requested symbol.

Parameters

- **scope** (*accepted arguments: ['5y','2y','1y','ytd','6m','3m','1m','next'], required*) – the range of data needed.
- **output** (*accepted values are ['dataframe', 'csv', 'excel'], optional*) – Determines the output of the data. Default is raw JSON.

get_financial_statements(*period=None*, *last=None*, *output=None*)

5,000 credits per symbol requested. This is useful if you want to update the query parameters after you've made some requests. It's just a time saver.

Returns

All financial statement data for the requested company and sets class attribute for each individual statement, and returns a list of the attribute names.

Parameters

- **period** (*accepted values are ['annual', 'quarterly'], optional*) – Time interval of statements. Defaults to quarterly
- **last** (*integer, optional*) – Number of periods to return, up to 4 for annual and 16 for quarterly. Defaults to 1.
- **output** (*accepted values are ['dataframe', 'csv', 'excel'], optional*) – Determines the output of the data. Default is raw JSON.

get_fund_ownership(*output=None*)

10,000 credits per symbol requested. Sets class attribute 'fund_ownership'.

Returns
10 largest institutional holders of the requested company.

Parameters
output (*accepted values are ['dataframe', 'csv', 'excel'], optional*) – Determines the output of the data. Default is raw JSON.

get_historical_price(*time_frame*, *date=None*, *chart_by_day=False*, *chart_close_only=False*, *output=None*)

10 credits per day requested when part of a time frame. (Full Data) 50 credits per single day, minute data.

Parameters

- **time_frame** (*string*) – Determines how far back to retrieve data. Set to 'date' if only one specific day is required.
- **date** (*string*) – If 'date' is specified in time_frame, this is the date that you wish to access.
- **chart_by_date** (*boolean*) – If a single date is requested, setting param to True only returns OHLC data instead of minutely data.
- **output** (*accepted values are ['dataframe', 'csv', 'excel'], optional*) – Determines the output of the data. Default is raw JSON.

get_income_statement(*period=None*, *last=None*, *output=None*)

1,000 credits per symbol requested.

Sets class attribute 'self.income_sheet'

Returns
Income sheet data for the requested company. If any non-JSON output is chosen, the method will return a Pandas DataFrame with the data from this endpoint.

Parameters

- **period** (*accepted values are ['annual', 'quarterly'], optional*) – Time interval of statements. Defaults to quarterly
- **last** (*integer, optional*) – Number of periods to return, up to 4 for annual and 16 for quarterly. Defaults to 1.
- **output** (*accepted values are ['dataframe', 'csv', 'excel'], optional*) – Determines the output of the data. Default is raw JSON.

get_insider_roster(*output=None*)

5,000 credits per symbol requested. Sets class attribute 'insider_roster'.

Returns
Top 10 insiders, with the most recent information.

Parameters
output (*accepted values are ['dataframe', 'csv', 'excel'], optional*) – Determines the output of the data. Default is raw JSON.

get_insider_transactions(*output=None*)

50 credits per transaction per symbol requested. Sets class attribute 'insider_transactions'.

Returns
Insider transactions with the most recent information.

Parameters
output (*accepted values are ['dataframe', 'csv', 'excel'], optional*) – Determines the output of the data. Default is raw JSON.

get_institutional_ownership(*output=None*)

10,000 credits per symbol requested. Sets class attribute 'institutional_ownership'.

Returns
10 largest instituional owners for the requested stock. This is defined as explicitly buy or sell-side only.

Parameters
output (*accepted values are ['dataframe', 'csv', 'excel'], optional*) – Determines the output of the data. Default is raw JSON.

get_key_stats(*stat=None*, *output=None*)

5 credits per symbol requested, 1 credit per stat per symbol requested. Sets class attribute 'key_stats'.

Returns
Important stats for the requested company.

Parameters

- **stat** (*string, optional*) – If you would like to querie one single stat, you can enter that here.
- **output** (*accepted values are ['dataframe', 'csv', 'excel'], optional*) – Determines the output of the data. Default is raw JSON.

get_largest_trades(*output=None*)

1 credit per trade per symbol requested. Sets class attribute 'largest_trades'.

Returns
15 minute delayed, last sale eligible trades.

Parameters

- **stat** (*string, optional*) – If you would like to querie one single stat, you can enter that here.
- **output** (*accepted values are ['dataframe', 'csv', 'excel'], optional*) – Determines the output of the data. Default is raw JSON.

get_logo()

1 credit per symbol requested. Sets class attribute 'logo'.

Returns
Google APIs link (bare url string) to the logo for the requested stock.

get_news(*last=10*, *output=None*)

1 credit per news article per symbol requested. Sets class attribute 'news'.

Returns
Intraday news from over 3,000 global news sources including major publications, regional media, and social.

Parameters

- **last** (*integer, min = 1 max = 50*) – Number of article to return. Defualt = 10
- **output** (*accepted values are ['dataframe', 'csv', 'excel'], optional*) – Determines the output of the data. Default is raw JSON.

get_ohlc(*output=None*)

2 credits per symbol requested. Sets class attribute 'ohlc'.

Returns
Official open and close for a give symbol.

Parameters
output (*accepted values are ['dataframe', 'csv', 'excel'], optional*) – Determines the output of the data. Default is raw JSON.

get_peers()

500 credits per symbol requested. Sets class attribute 'peers'.

Returns
List of peer company's tickers in Python list form.

get_price_target(*output=None*)

Premium Data. 500 premium credits per symbol requested. CSV is formatted vertically with keys in the first column. Sets class attribute 'price_target'.

Returns
Latest avg, high, and low analyst price target for a symbol.

Parameters
output (*accepted values are ['dataframe', 'csv', 'excel'], optional*) – Determines the output of the data. Default is raw JSON.

get_quote(*output=None*)

1 credit per symbol requested. CSV is formatted vertically with keys in the first column. Sets class attribute 'quote'.

Returns
Quote data for the requested symbol.

Parameters
output (*accepted values are ['dataframe', 'csv', 'excel'], optional*) – Determines the output of the data. Default is raw JSON.

pandas_financial_json(*json*, *statement*)

Returns a dataframe for the requested financial statement.

Parameters

- **json** (*string, raw json, required*) – the raw json output from IEX Cloud containing financial statement data.
- **statement** (*accepted values are : 'balancesheet', 'incomestatement', and 'cashflow'.*) – the name of the statement as used by IEX Cloud.

pandas_listofdict_json(*json*)

Returns a dataframe for the requested list of dictionaries json document.

Parameters

json (*raw json*) – the raw json output from IEX Cloud containing a list of dictionary.

pandas_price_json(*json*)

Returns a dataframe for the requested price statement.

Parameters

json (*raw json*) – the raw json output from IEX Cloud containing daily price data.

pandas_singledict_json(*json*, *in_list=False*)

Returns a dataframe for the requested dictionary json document.

Parameters

- **json** (*raw json*) – the raw json output from IEX Cloud containing a list of dictionary.
- **in_list** (*boolean*) – whether the disctionary is conatined within a list element, that is, inside square braces.

pickle_class_state()

This method takes the current state of the class and pickles it as-is. This will preserve all attributes of the current class instance. Recalling/loading the class can be done with the `iex.unpickle_class_state()` Filename is automatically created as '#ticker#_#date#_pickle.pickle'

price()

1 credit per symbol requested.

Returns

Most recent price for the requested company and sets class attribute 'self.price'.

price_information()

6 credits per symbol requested. Makes requests to the key stat and price endpoints. Sets class attributes for:

52 week high (week52_high), 52 week low (week52_low)

200 day moving average (moving_average_200), 50 day moving average (moving_average_50)

Most recent price (self.price)

set_date()

Sets the date attribute. This is needed keep save and load funcionality smooth.

4.2 IEXMarket Class

class iex.**IEXMarket**

get_all_commodity_data(*output_csv=False*)

10,000 credits per request per day. Returns all available current commodity datapoints.

Parameters

output (*boolean, optional*) – Determines the output of the data. Default is raw JSON.

get_all_economic_data(*output_csv=False*)

18,000 credits per request per day. :return: All available current economic indicator datapoints.

Parameters

output (*boolean, optional*) – Determines the output of the data. Default is raw JSON.

get_market_datapoint(*symbol*, *key='market'*, ***queries*)

1,000 credits per symbol requested per date. Sets a class attribute equal to the symbol requested.

Class attribute 'available_economic_symbols' is a dictionary of available economic symbols and their description. Class attribute 'available_commodity_symbols' is a dictionary of available commodity symbols and their description

Returns

Various economic and commodity values as a datapoint.

Parameters

- **symbol** (*string, required*) – The symbol of the economic indicator or commodity datapoint requested.
- ****queries** – Additional query parameters you want to include as listed in IEX Docs.

CHAPTER

FIVE

POLYGON.IO MODULES

5.1 Stocks Endpoint

`polygon.stocks.aggregates(stock_ticker, multiplier, timespan, from_date, to_date, external=False, **query_params)`

Returns

Get aggregate bars for a stock over a given date range in custom time window sizes.

Parameters

- **stock_ticker** (*required, string*) – The ticker symbol of the stock.
- **multiplier** (*required, integer*) – The size of the timespan multiplier.
- **timespan** (*required, string, [minute, hour, day, week, month, quarter, year]*) – The size of the time window.
- **from** (*required, string, date ex. '2021-07-22'*) – The start of the aggregate time window.
- **to** (*required, string, date ex. '2021-07-22'*) – The end of the aggregate time window.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **adjusted** (*optional, boolean, default True*) – Whether or not the results are adjusted for splits. Set this to false to get results that are NOT adjusted for splits.
- **sort** (*optional, string, [desc, asc]*) – Sort the results by timestamp.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.

`polygon.stocks.daily_open_close(stock_ticker, date, external=False, **query_params)`

Returns

Get the open, close and afterhours prices of a stock symbol on a certain date.

Parameters

- **stock_ticker** (*required, string*) – The ticker symbol of the stock.
- **date** (*required, string, date ex. '2021-07-22'*) – The beginning date for the aggregate window.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.

- **adjusted** (*optional, boolean, default True*) – Whether or not the results are adjusted for splits. Set this to false to get results that are NOT adjusted for splits.

`polygon.stocks.dividends_v3(external=False, **query_params)`

Returns

Get a list of historical cash dividends, including the ticker symbol, declaration date, ex-dividend date, record date, pay date, frequency, and amount.

Parameters

- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **stock_ticker** (*required, string*) – The ticker symbol of the stock.
- **ex_dividend_date** (*optional, string, date ex. '2021-07-22'*) – Query by ex-dividend date with the format YYYY-MM-DD.
- **record_date** (*optional, string, date ex. '2021-07-22'*) – Query by record date with the format YYYY-MM-DD.
- **declaration_date** (*optional, string, date ex. '2021-07-22'*) – Query by declaration date with the format YYYY-MM-DD.
- **pay_date** (*optional, string, date ex. '2021-07-22'*) – Query by pay date with the format YYYY-MM-DD.
- **frequency** (*optional, integer, eg. [0,1,2,4,12]*) – Query by the number of times per year the dividend is paid out.
- **cash_amount** (*optional, float*) – Query by the cash amount of the dividend.
- **dividend_type** (*optional, string, Consistent Dividends = CD, Special Cash Dividends = [SC]*) – Query by the type of dividend.
- **order** (*optional, string, [desc, asc]*) – Order results based on the sort field.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.
- **sort** (*optional, string*) – Sort field used for ordering.

`polygon.stocks.financials(external=False, **query_params)`

This API is experimental. :return: Get historical financial data for a stock ticker.

Parameters

- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **stock_ticker** (*required, string*) – The ticker symbol of the stock.
- **cik** –
- **company_name** –
- **sic** –

:param : :type : :param : :type : :param : :type : :param : :type : :param order: Order results based on the sort field. :type order: optional, string, [desc, asc] :param limit: Limits the number of base aggregates queried to create the aggregate results. :type limit: optional, integer, default 5000, max 50000 :param sort: Sort field used for ordering. :type sort: optional, string

polygon.stocks.**grouped_daily**(*date*, *external=False*, ***query_params*)

Returns
Get the daily open, high, low, and close (OHLC) for the entire stocks/equities markets.

Parameters

- **date** (*required, string, date ex. '2021-07-22'*) – The beginning date for the aggregate window.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **adjusted** (*optional, boolean, default True*) – Whether or not the results are adjusted for splits. Set this to false to get results that are NOT adjusted for splits.

polygon.stocks.**last_quote**(*stock_ticker*, *external=False*, ***query_params*)

Returns
Get the most recent NBBO (Quote) tick for a given stock.

Parameters

- **stock_ticker** (*required, string*) – The ticker symbol of the stock.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.

polygon.stocks.**last_trade**(*stock_ticker*, *external=False*, ***query_params*)

Returns
Get the most recent trade for a given stock.

Parameters
stock_ticker (*required, string*) – The ticker symbol of the stock.

polygon.stocks.**previous_close**(*stock_ticker*, *external=False*, ***query_params*)

Returns
Get the previous day's open, high, low, and close (OHLC) for the specified stock ticker.

Parameters

- **stock_ticker** (*required, string*) – The ticker symbol of the stock.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **adjusted** (*optional, boolean, default True*) – Whether or not the results are adjusted for splits. Set this to false to get results that are NOT adjusted for splits.

polygon.stocks.**quotes_v2**(*stock_ticker*, *external=False*, ***query_params*)

Returns
Get NBBO quotes for a given ticker symbol on a specified date.

Parameters

- **stock_ticker** (*required, string*) – The ticker symbol of the stock.
- **date** (*required, string, date ex. '2021-07-22'*) – The beginning date for the aggregate window.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.

- **timestamp** (*optional, string, date or nanosecond timestamp*) – Query by timestamp. Either a date with the format YYYY-MM-DD or a nanosecond timestamp.
- **timestamp_limit** (*optional, integer*) – The maximum timestamps allowed in the results.
- **reverse** (*optional, boolean*) – Reverse the order of the results.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.

polygon.stocks.**quotes_v3**(*stock_ticker*, *external=False*, ***query_params*)

Returns
Get NBBO quotes for a ticker symbol in a given time range.

Parameters

- **stock_ticker** (*required, string*) – The ticker symbol of the stock.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **timestamp** (*optional, string, date or nanosecond timestamp*) – Query by timestamp. Either a date with the format YYYY-MM-DD or a nanosecond timestamp.
- **order** (*optional, string, [desc, asc]*) – Order results based on the sort field.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.
- **sort** (*optional, string*) – Sort field used for ordering.

polygon.stocks.**snapshot_all**(*external=False*, ***query_params*)

Returns
Get the most up-to-date market data for all traded stock symbols.

Parameters

- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **tickers** (*optional, string, list of tickers*) – A comma separated list of tickers to get snapshots for.

polygon.stocks.**snapshot_gain_lose**(*direction*, *external=False*, ***query_params*)

Returns
Get the most up-to-date market data for the current top 20 gainers or losers of the day in the stocks/equities markets.

Parameters

- **direction** (*required, string, [gainers, losers]*) – The direction of the snapshot results to return.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.

polygon.stocks.**snapshot_ticker**(*stock_ticker*, *external=False*, ***query_params*)

Returns
Get the most up-to-date market data for a single traded stock ticker.

Parameters

- **stock_ticker** (*required, string*) – The ticker symbol of the stock.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.

`polygon.stocks.stock_splits_v3(external=False, **query_params)`

Returns
Get a list of historical stock splits, including the ticker symbol, the execution date, and the factors of the split ratio.

Parameters

- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **stock_ticker** (*required, string*) – The ticker symbol of the stock.
- **execution_date** (*optional, string, date ex. '2021-07-22'*) – Query by execution date.

:param reverse_split:Query for reverse splits only. :type reverse_split: optional, boolean :param order: Order results based on the sort field. :type order: optional, string, [desc, asc] :param limit: Limits the number of base aggregates queried to create the aggregate results. :type limit: optional, integer, default 5000, max 50000 :param sort: Sort field used for ordering. :type sort: optional, string

`polygon.stocks.ticker_details(stock_ticker, external=False, **query_params)`

Returns
Get a single ticker supported by Polygon.io. This response will have detailed information about the ticker and the company behind it.

Parameters

- **stock_ticker** (*required, string*) – The ticker symbol of the stock.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **date** (*optional, string, date ex. '2021-07-22'*) – The beginning date for the aggregate window.

`polygon.stocks.trades(stock_ticker, date, external=False, **query_params)`

Returns
Get trades for a given ticker symbol on a specified date.

Parameters

- **stock_ticker** (*required, string*) – The ticker symbol of the stock.
- **date** (*required, string, date ex. '2021-07-22'*) – The beginning date for the aggregate window.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **timestamp** (*optional, string, date or nanosecond timestamp*) – Query by timestamp. Either a date with the format YYYY-MM-DD or a nanosecond timestamp.
- **timestamp_limit** (*optional, integer*) – The maximum timestamps allowed in the results.

- **reverse** (*optional, boolean*) – Reverse the order of the results.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.

`polygon.stocks.trades_v3(stock_ticker, external=False, **query_params)`

Returns
Get trades for a ticker symbol in a given time range.

Parameters

- **stock_ticker** (*required, string*) – The ticker symbol of the stock.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **timestamp** (*optional, string, date or nanosecond timestamp*) – Query by timestamp. Either a date with the format YYYY-MM-DD or a nanosecond timestamp.
- **order** (*optional, string, [desc, asc]*) – Order results based on the sort field.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.
- **sort** (*optional, string*) – Sort field used for ordering.

5.2 Options Endpoint

`polygon.options.aggregates(options_ticker, multiplier, timespan, from_date, to_date, external=False, **query_params)`

Returns
Get aggregate bars for an option contract over a given date range in custom time window sizes.

Parameters

- **options_ticker** (*required, string*) – The ticker symbol of the options contract.
- **multiplier** (*required, integer*) – The size of the timespan multiplier.
- **timespan** (*required, string, [minute, hour, day, week, month, quarter, year]*) – The size of the time window.
- **from** (*required, string, date ex. '2021-07-22'*) – The start of the aggregate time window.
- **to** (*required, string, date ex. '2021-07-22'*) – The end of the aggregate time window.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **adjusted** (*optional, boolean, default True*) – Whether or not the results are adjusted for splits. Set this to false to get results that are NOT adjusted for splits.
- **sort** (*optional, string, [desc, asc]*) – Sort the results by timestamp.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.

`polygon.options.daily_open_close(options_ticker, date, external=False, **query_params)`

Returns
Get the open, close and afterhours prices of an options contract on a certain date.

Parameters

- **options_ticker** (*required, string*) – The ticker symbol of the options contract.
- **date** (*required, string, date ex. '2021-07-22'*) – The beginning date for the aggregate window.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **adjusted** (*optional, boolean, default True*) – Whether or not the results are adjusted for splits. Set this to false to get results that are NOT adjusted for splits.

`polygon.options.last_trade(options_ticker, external=False, **query_params)`

Returns
Get the most recent trade for a given options contract.

Parameters

- **options_ticker** (*required, string*) – The ticker symbol of the options contract.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.

`polygon.options.options_contract(options_ticker, external=False, **query_params)`

Returns
Get an options contract.

Parameters

- **options_ticker** (*required, string*) – The ticker symbol of the options contract.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **as_of** (*optional, string, date ex. '2021-07-22', default is today*) – Specify a point in time for the contract as of this date with format YYYY-MM-DD

`polygon.options.options_contracts(external=False, **query_params)`

Returns
Query for historical options contracts. This provides both active and expired options contracts.

Parameters

- **underlying_asset** (*required, string*) – The underlying ticker symbol of the option contract.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **contract_type** (*optional, string*) – Query by the type of contract.
- **expiration_date** (*optional, string, date ex. '2021-07-22'*) – Query by contract expiration with date format YYYY-MM-DD.
- **as_of** (*optional, string, date ex. '2021-07-22', default is today*) – Specify a point in time for the contract as of this date with format YYYY-MM-DD

- **strike_price** (*optional, float*) – Query by strike price of a contract.
- **expired** (*optional, boolean, default false*) – Query for expired contracts
- **order** (*optional, string, [desc, asc]*) – Order results based on the sort field.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.
- **sort** (*optional, string*) – Sort field used for ordering.

polygon.options.**previous_close**(*options_ticker*, *external=False*, ***query_params*)

Returns
Get the previous day's open, high, low, and close (OHLC) for the specified option contract.

Parameters

- **options_ticker** (*required, string*) – The ticker symbol of the options contract.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **adjusted** (*optional, boolean, default True*) – Whether or not the results are adjusted for splits. Set this to false to get results that are NOT adjusted for splits.

polygon.options.**quotes**(*options_ticker*, *external=False*, ***query_params*)

Returns
Get quotes for an options ticker symbol in a given time range.

Parameters

- **options_ticker** (*required, string*) – The ticker symbol of the options contract.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **timestamp** (*optional, string, date or nanosecond timestamp*) – Query by timestamp. Either a date with the format YYYY-MM-DD or a nanosecond timestamp.
- **order** (*optional, string, [desc, asc]*) – Order results based on the sort field.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.
- **sort** (*optional, string*) – Sort field used for ordering.

polygon.options.**snapshot_options**(*underlying_asset*, *option_contract*, *external=False*, ***query_params*)

Returns
Get the snapshot of an option contract for a stock equity.

Parameters

- **underlying_asset** (*required, string*) – The underlying ticker symbol of the option contract.
- **option_contract** (*required, string*) – The option contract identifier.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.

`polygon.options.trades(options_ticker, external=False, **query_params)`

Returns
Get trades for an options ticker symbol in a given time range.

Parameters

- **options_ticker** (*required, string*) – The ticker symbol of the options contract.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **timestamp** (*optional, string, date or nanosecond timestamp*) – Query by timestamp. Either a date with the format YYYY-MM-DD or a nanosecond timestamp.
- **order** (*optional, string, [desc, asc]*) – Order results based on the sort field.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.
- **sort** (*optional, string*) – Sort field used for ordering.

5.3 Forex Endpoint

`polygon.forex.aggregates(forex_ticker, multiplier, timespan, from_date, to_date, external=False, **query_params)`

Returns
Get aggregate bars for a forex pair over a given date range in custom time window sizes.

Parameters

- **forex_ticker** (*required, string*) – The ticker symbol of the currency pair.
- **multiplier** (*required, integer*) – The size of the timespan multiplier.
- **timespan** (*required, string, [minute, hour, day, week, month, quarter, year]*) – The size of the time window.
- **from_date** (*required, string, date ex. '2021-07-22'*) – The start of the aggregate time window.
- **to** (*required, string, date ex. '2021-07-22'*) – The end of the aggregate time window.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **adjusted** (*optional, boolean, default True*) – Whether or not the results are adjusted for splits. Set this to false to get results that are NOT adjusted for splits.
- **sort** (*optional, string, [desc, asc]*) – Sort the results by timestamp.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.

`polygon.forex.conversion(from_date, to, external=False, **query_params)`

Returns
Get currency conversions using the latest market conversion rates. Note than you can convert in both directions.

Parameters

- **from_date** (*required, string, date ex. '2021-07-22'*) – The start of the aggregate time window.
- **to** (*required, string, date ex. '2021-07-22'*) – The end of the aggregate time window.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **amount** (*optional, integer*) – The amount to convert, with a decimal.
- **precision** (*optional, integer*) – The decimal precision of the conversion. Defaults to 2 which is 2 decimal places accuracy.

polygon.forex.**grouped_daily**(*date*, *external=False*, ***query_params*)

Returns

Get the daily open, high, low, and close (OHLC) for the entire forex markets.

Parameters

- **date** (*required, string, date ex. '2021-07-22'*) – The beginning date for the aggregate window.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **adjusted** (*optional, boolean, default True*) – Whether or not the results are adjusted for splits. Set this to false to get results that are NOT adjusted for splits.

polygon.forex.**historic_ticks**(*from_date*, *to*, *date*, *external=False*, ***query_params*)

Returns

Get historic ticks for a forex currency pair.

Parameters

- **from_date** (*required, string, date ex. '2021-07-22'*) – The start of the aggregate time window.
- **to** (*required, string, date ex. '2021-07-22'*) – The end of the aggregate time window.
- **date** (*required, string, date ex. '2021-07-22'*) – The beginning date for the aggregate window.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **offset** (*optional, string, date or nanosecond timestamp*) – The timestamp offset, used for pagination. This is the offset at which to start the results.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.

polygon.forex.**last_pair_quote**(*from_date*, *to*, *external=False*, ***query_params*)

Returns

Get the last quote tick for a forex currency pair.

Parameters

- **from_date** (*required, string, date ex. '2021-07-22'*) – The start of the aggregate time window.
- **to** (*required, string, date ex. '2021-07-22'*) – The end of the aggregate time window.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.

polygon.forex.**previous_close**(*forex_ticker*, *external=False*, ***query_params*)

Returns
Get the previous day's open, high, low, and close (OHLC) for the specified forex pair.

Parameters

- **forex_ticker** (*required, string*) – The ticker symbol of the currency pair.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **adjusted** (*optional, boolean, default True*) – Whether or not the results are adjusted for splits. Set this to false to get results that are NOT adjusted for splits.

polygon.forex.**quotes**(*forex_ticker*, *external=False*, ***query_params*)

Returns
Get BBO quotes for a ticker symbol in a given time range.

Parameters

- **forex_ticker** (*required, string*) – The ticker symbol of the currency pair.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **timestamp** (*optional, string, date or nanosecond timestamp*) – Query by timestamp. Either a date with the format YYYY-MM-DD or a nanosecond timestamp.
- **order** (*optional, string, [desc, asc]*) – Order results based on the sort field.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.
- **sort** (*optional, string*) – Sort field used for ordering.

polygon.forex.**snapshot_all**(*external=False*, ***query_params*)

Returns
Get the current minute, day, and previous day's aggregate, as well as the last trade and quote for all traded forex symbols.

Parameters

- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **tickers** (*optional, string, list of tickers*) – A comma separated list of tickers to get snapshots for.

polygon.forex.**snapshot_gain_lose**(*direction*, *external=False*, ***query_params*)

Returns
Get the current top 20 gainers or losers of the day in forex markets.

Parameters

- **direction** (*required, string, [gainers, losers]*) – The direction of the snapshot results to return.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.

polygon.forex.**snapshot_ticker**(*forex_ticker*, *external=False*, ***query_params*)

:return:Get the current minute, day, and previous day's aggregate, as well as the last trade and quote for a single traded currency symbol.

Parameters

- **forex_ticker** (*required, string*) – The ticker symbol of the currency pair.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.

5.4 Reference Endpoint

polygon.reference.**conditions**(*external=False*, ***query_params*)

Returns
List all conditions that Polygon.io uses.

Parameters

- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **asset_class** (*optional, string*) – The asset class by which you would like to filter results.
- **data_type** (*optional, string*) – The data type by which you would like to filter results.
- **id** (*optional, integer*) – Filter for conditions with a given ID.
- **sip** (*optional, string, [CTA, UTP, OPRA]*) – Filter by SIP. If the condition contains a mapping for that SIP, the condition will be returned.
- **order** (*optional, string, [desc, asc]*) – Order results based on the sort field.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.
- **sort** (*optional, string*) – Sort field used for ordering.

polygon.reference.**exchanges**(*external=False*, ***query_params*)

Returns
List all exchanges that Polygon.io knows about.

Parameters

- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **asset_class** (*optional, string*) – The asset class by which you would like to filter results.
- **locale** (*optional, string*) – The locale by which you would like to filter results.

polygon.reference.**market_holidays**(*external=False*, ***query_params*)

Returns
Get upcoming market holidays and their open/close times.

Parameters
external (*optional, string*) – Your Polygon API key if you are not using environment variables.

polygon.reference.**market_status**(*external=False*, ***query_params*)

Returns
Get the current trading status of the exchanges and overall financial markets.

Parameters
external (*optional, string*) – Your Polygon API key if you are not using environment variables.

polygon.reference.**ticker_news**(*external=False*, ***query_params*)

Returns
Get the most recent news articles relating to a stock ticker symbol, including a summary of the article and a link to the original source.

Parameters

- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **ticker** (*optional, string*) – The ticker for which you would like to query information for.
- **published_utc** (*optional, string, date ex. '2021-07-22'*) – Return results published on, before, or after this date.
- **order** (*optional, string, [desc, asc]*) – Order results based on the sort field.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.
- **sort** (*optional, string*) – Sort field used for ordering.

polygon.reference.**ticker_types**(*external=False*, ***query_params*)

Returns
List all ticker types that Polygon.io has.

Parameters

- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **asset_class** (*optional, string*) – The asset class by which you would like to filter results.
- **locale** (*optional, string*) – The locale by which you would like to filter results.

polygon.reference.**tickers**(*external=False*, ***query_params*)

Returns
Query all ticker symbols which are supported by Polygon.io.

Parameters

- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **ticker** (*optional, string*) – The ticker for which you would like to query information for.
- **type** (*optional, string, default is to query all types*) – The type of the ticker, Available through Polygon.io's Ticker Types API.
- **market** (*optional, string, default is to query all markets*) – The market you would like to filter results by.
- **exchange** (*optional, string, default is to query all exchanges*) – The primary exchange of the asset in the ISO code format.
- **cusip** (*optional, string, default is to query all CUSIPs*) – The CUSIP code of the asset you want to search for.
- **cik** (*optional, string, default is to query all CIKs*) – The CIK of the asset you want to search for.
- **date** (*optional, string, date ex. '2021-07-22', default is most recent date*) – Specify a point in time to retrieve tickers available on that date.
- **search** (*optional, string*) – Search for terms within the ticker and/or company name.
- **active** (*optional, boolean, default is true*) – Specify if the tickers returned should be actively traded on the queried date
- **sort** (*optional, string*) – Sort field used for ordering.
- **order** (*optional, string, [desc, asc]*) – Order results based on the sort field.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.

CHAPTER

SIX

POLYGON.IO CLASSES

6.1 polygonStocks Class

class polygon.**polygonStocks**(*ticker*, *external=None*)

get_aggregates(*timespan*, *from_date*, *to_date*, ***query_params*)

Returns
Get aggregate bars for a stock over a given date range in custom time window sizes.

Parameters

- **stock_ticker** (*required, string*) – The ticker symbol of the stock.
- **multiplier** (*required, integer*) – The size of the timespan multiplier.
- **timespan** (*required, string, [minute, hour, day, week, month, quarter, year]*) – The size of the time window.
- **from** (*required, string, date ex. '2021-07-22'*) – The start of the aggregate time window.
- **to** (*required, string, date ex. '2021-07-22'*) – The end of the aggregate time window.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **adjusted** (*optional, boolean, default True*) – Whether or not the results are adjusted for splits. Set this to false to get results that are NOT adjusted for splits.
- **sort** (*optional, string, [desc, asc]*) – Sort the results by timestamp.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.

get_daily_open_close(*date*, ***query_params*)

Returns
Get the open, close and afterhours prices of a stock symbol on a certain date.

Parameters

- **stock_ticker** (*required, string*) – The ticker symbol of the stock.
- **date** (*required, string, date ex. '2021-07-22'*) – The beginning date for the aggregate window.

- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **adjusted** (*optional, boolean, default True*) – Whether or not the results are adjusted for splits. Set this to false to get results that are NOT adjusted for splits.

get_dividends_v3(***query_params*)

Returns

Get a list of historical cash dividends, including the ticker symbol, declaration date, ex-dividend date, record date, pay date, frequency, and amount.

Parameters

- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **stock_ticker** (*required, string*) – The ticker symbol of the stock.
- **ex_dividend_date** (*optional, string, date ex. '2021-07-22'*) – Query by ex-dividend date with the format YYYY-MM-DD.
- **record_date** (*optional, string, date ex. '2021-07-22'*) – Query by record date with the format YYYY-MM-DD.
- **declaration_date** (*optional, string, date ex. '2021-07-22'*) – Query by declaration date with the format YYYY-MM-DD.
- **pay_date** (*optional, string, date ex. '2021-07-22'*) – Query by pay date with the format YYYY-MM-DD.
- **frequency** (*optional, integer, eg. [0,1,2,4,12]*) – Query by the number of times per year the dividend is paid out.
- **cash_amount** (*optional, float*) – Query by the cash amount of the dividend.
- **dividend_type** (*optional, string, Consistent Dividends = CD, Special Cash Dividends = [SC]*) – Query by the type of dividend.
- **order** (*optional, string, [desc, asc]*) – Order results based on the sort field.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.
- **sort** (*optional, string*) – Sort field used for ordering.

get_financials(***query_params*)

This API is experimental. :return: Get historical financial data for a stock ticker.

Parameters

- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **stock_ticker** (*required, string*) – The ticker symbol of the stock.
- **cik** –
- **company_name** –
- **sic** –

:param : :type : :param : :type : :param : :type : :param : :type : :param order: Order results based on the sort field. :type order: optional, string, [desc, asc] :param limit: Limits the number of base aggregates queried to create the aggregate results. :type limit: optional, integer, default 5000, max 50000 :param sort: Sort field used for ordering. :type sort: optional, string

get_grouped_daily(*date*, ***query_params*)

Returns
Get the daily open, high, low, and close (OHLC) for the entire stocks/equities markets.

Parameters

- **date** (*required, string, date ex. '2021-07-22'*) – The beginning date for the aggregate window.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **adjusted** (*optional, boolean, default True*) – Whether or not the results are adjusted for splits. Set this to false to get results that are NOT adjusted for splits.

get_last_quote(***query_params*)

Returns
Get the most recent NBBO (Quote) tick for a given stock.

Parameters

- **stock_ticker** (*required, string*) – The ticker symbol of the stock.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.

get_last_trade(***query_params*)

Returns
Get the most recent trade for a given stock.

Parameters
stock_ticker (*required, string*) – The ticker symbol of the stock.

get_previous_close(***query_params*)

Returns
Get the previous day's open, high, low, and close (OHLC) for the specified stock ticker.

Parameters

- **stock_ticker** (*required, string*) – The ticker symbol of the stock.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **adjusted** (*optional, boolean, default True*) – Whether or not the results are adjusted for splits. Set this to false to get results that are NOT adjusted for splits.

get_quotes_v2(***query_params*)

Returns
Get NBBO quotes for a given ticker symbol on a specified date.

Parameters

- **stock_ticker** (*required, string*) – The ticker symbol of the stock.
- **date** (*required, string, date ex. '2021-07-22'*) – The beginning date for the aggregate window.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.

- **timestamp** (*optional, string, date or nanosecond timestamp*) – Query by timestamp. Either a date with the format YYYY-MM-DD or a nanosecond timestamp.
- **timestamp_limit** (*optional, integer*) – The maximum timestamps allowed in the results.
- **reverse** (*optional, boolean*) – Reverse the order of the results.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.

get_quotes_v3(***query_params*)

Returns
Get NBBO quotes for a ticker symbol in a given time range.

Parameters

- **stock_ticker** (*required, string*) – The ticker symbol of the stock.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **timestamp** (*optional, string, date or nanosecond timestamp*) – Query by timestamp. Either a date with the format YYYY-MM-DD or a nanosecond timestamp.
- **order** (*optional, string, [desc, asc]*) – Order results based on the sort field.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.
- **sort** (*optional, string*) – Sort field used for ordering.

get_snapshot_all(***query_params*)

Returns
Get the most up-to-date market data for all traded stock symbols.

Parameters

- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **tickers** (*optional, string, list of tickers*) – A comma separated list of tickers to get snapshots for.

get_snapshot_gain_lose(*direction*, ***query_params*)

Returns
Get the most up-to-date market data for the current top 20 gainers or losers of the day in the stocks/equities markets.

Parameters

- **direction** (*required, string, [gainers, losers]*) – The direction of the snapshot results to return.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.

get_snapshot_ticker(***query_params*)

Returns
Get the most up-to-date market data for a single traded stock ticker.

Parameters

- **stock_ticker** (*required, string*) – The ticker symbol of the stock.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.

get_stock_splits_v3(***query_params*)

Returns
Get a list of historical stock splits, including the ticker symbol, the execution date, and the factors of the split ratio.

Parameters

- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **stock_ticker** (*required, string*) – The ticker symbol of the stock.
- **execution_date** (*optional, string, date ex. '2021-07-22'*) – Query by execution date.

:param reverse_split:Query for reverse splits only. :type reverse_split: optional, boolean :param order: Order results based on the sort field. :type order: optional, string, [desc, asc] :param limit: Limits the number of base aggregates queried to create the aggregate results. :type limit: optional, integer, default 5000, max 50000 :param sort: Sort field used for ordering. :type sort: optional, string

get_ticker_details(***query_params*)

Returns
Get a single ticker supported by Polygon.io. This response will have detailed information about the ticker and the company behind it.

Parameters

- **stock_ticker** (*required, string*) – The ticker symbol of the stock.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **date** (*optional, string, date ex. '2021-07-22'*) – The beginning date for the aggregate window.

get_trades(*date*, ***query_params*)

Returns
Get trades for a given ticker symbol on a specified date.

Parameters

- **stock_ticker** (*required, string*) – The ticker symbol of the stock.
- **date** (*required, string, date ex. '2021-07-22'*) – The beginning date for the aggregate window.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **timestamp** (*optional, string, date or nanosecond timestamp*) – Query by timestamp. Either a date with the format YYYY-MM-DD or a nanosecond timestamp.
- **timestamp_limit** (*optional, integer*) – The maximum timestamps allowed in the results.

- **reverse** (*optional, boolean*) – Reverse the order of the results.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.

get_trades_v3(***query_params*)

Returns
Get trades for a ticker symbol in a given time range.

Parameters

- **stock_ticker** (*required, string*) – The ticker symbol of the stock.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **timestamp** (*optional, string, date or nanosecond timestamp*) – Query by timestamp. Either a date with the format YYYY-MM-DD or a nanosecond timestamp.
- **order** (*optional, string, [desc, asc]*) – Order results based on the sort field.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.
- **sort** (*optional, string*) – Sort field used for ordering.

6.2 polygonOptions Class

class polygon.**polygonOptions**(*ticker*, *external=None*)

get_aggregates(*multiplier*, *timespan*, *from_date*, *to_date*, ***query_params*)

Returns
Get aggregate bars for an option contract over a given date range in custom time window sizes.

Parameters

- **options_ticker** (*required, string*) – The ticker symbol of the options contract.
- **multiplier** (*required, integer*) – The size of the timespan multiplier.
- **timespan** (*required, string, [minute, hour, day, week, month, quarter, year]*) – The size of the time window.
- **from** (*required, string, date ex. '2021-07-22'*) – The start of the aggregate time window.
- **to** (*required, string, date ex. '2021-07-22'*) – The end of the aggregate time window.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **adjusted** (*optional, boolean, default True*) – Whether or not the results are adjusted for splits. Set this to false to get results that are NOT adjusted for splits.
- **sort** (*optional, string, [desc, asc]*) – Sort the results by timestamp.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.

`get_daily_open_close`(*date*, ***query_params*)

Returns
Get the open, close and afterhours prices of an options contract on a certain date.

Parameters

- **options_ticker** (*required, string*) – The ticker symbol of the options contract.
- **date** (*required, string, date ex. '2021-07-22'*) – The beginning date for the aggregate window.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **adjusted** (*optional, boolean, default True*) – Whether or not the results are adjusted for splits. Set this to false to get results that are NOT adjusted for splits.

`get_last_trade`(***query_params*)

Returns
Get the most recent trade for a given options contract.

Parameters

- **options_ticker** (*required, string*) – The ticker symbol of the options contract.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.

`get_options_contract`(***query_params*)

Returns
Get an options contract.

Parameters

- **options_ticker** (*required, string*) – The ticker symbol of the options contract.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **as_of** (*optional, string, date ex. '2021-07-22', default is today*) – Specify a point in time for the contract as of this date with format YYYY-MM-DD

`get_options_contracts`(***query_params*)

Returns
Query for historical options contracts. This provides both active and expired options contracts.

Parameters

- **underlying_asset** (*required, string*) – The underlying ticker symbol of the option contract.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **contract_type** (*optional, string*) – Query by the type of contract.
- **expiration_date** (*optional, string, date ex. '2021-07-22'*) – Query by contract expiration with date format YYYY-MM-DD.
- **as_of** (*optional, string, date ex. '2021-07-22', default is today*) – Specify a point in time for the contract as of this date with format YYYY-MM-DD

- **strike_price** (*optional, float*) – Query by strike price of a contract.
- **expired** (*optional, boolean, default false*) – Query for expired contracts
- **order** (*optional, string, [desc, asc]*) – Order results based on the sort field.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.
- **sort** (*optional, string*) – Sort field used for ordering.

get_previous_close(***query_params*)

Returns
Get the previous day's open, high, low, and close (OHLC) for the specified option contract.

Parameters

- **options_ticker** (*required, string*) – The ticker symbol of the options contract.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **adjusted** (*optional, boolean, default True*) – Whether or not the results are adjusted for splits. Set this to false to get results that are NOT adjusted for splits.

get_quotes(***query_params*)

Returns
Get quotes for an options ticker symbol in a given time range.

Parameters

- **options_ticker** (*required, string*) – The ticker symbol of the options contract.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **timestamp** (*optional, string, date or nanosecond timestamp*) – Query by timestamp. Either a date with the format YYYY-MM-DD or a nanosecond timestamp.
- **order** (*optional, string, [desc, asc]*) – Order results based on the sort field.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.
- **sort** (*optional, string*) – Sort field used for ordering.

get_snapshot_options(*underlying_asset*, *option_contract*, ***query_params*)

Returns
Get the snapshot of an option contract for a stock equity.

Parameters

- **underlying_asset** (*required, string*) – The underlying ticker symbol of the option contract.
- **option_contract** (*required, string*) – The option contract identifier.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.

get_trades(***query_params*)

Returns
Get trades for an options ticker symbol in a given time range.

Parameters

- **options_ticker** (*required, string*) – The ticker symbol of the options contract.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **timestamp** (*optional, string, date or nanosecond timestamp*) – Query by timestamp. Either a date with the format YYYY-MM-DD or a nanosecond timestamp.
- **order** (*optional, string, [desc, asc]*) – Order results based on the sort field.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.
- **sort** (*optional, string*) – Sort field used for ordering.

6.3 polygonForex Class

class polygon.**polygonForex**(*ticker*, *external=None*)

get_aggregates(*multiplier*, *timespan*, *from_date*, *to_date*, ***query_params*)

Returns
Get aggregate bars for a forex pair over a given date range in custom time window sizes.

Parameters

- **forex_ticker** (*required, string*) – The ticker symbol of the currency pair.
- **multiplier** (*required, integer*) – The size of the timespan multiplier.
- **timespan** (*required, string, [minute, hour, day, week, month, quarter, year]*) – The size of the time window.
- **from_date** (*required, string, date ex. '2021-07-22'*) – The start of the aggregate time window.
- **to** (*required, string, date ex. '2021-07-22'*) – The end of the aggregate time window.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **adjusted** (*optional, boolean, default True*) – Whether or not the results are adjusted for splits. Set this to false to get results that are NOT adjusted for splits.
- **sort** (*optional, string, [desc, asc]*) – Sort the results by timestamp.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.

get_conversion(*from_date*, *to_date*, ***query_params*)

Returns
Get currency conversions using the latest market conversion rates. Note than you can convert in both directions.

Parameters

- **from_date** (*required, string, date ex. '2021-07-22'*) – The start of the aggregate time window.
- **to** (*required, string, date ex. '2021-07-22'*) – The end of the aggregate time window.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **amount** (*optional, integer*) – The amount to convert, with a decimal.
- **precision** (*optional, integer*) – The decimal precision of the conversion. Defaults to 2 which is 2 decimal places accuracy.

get_historic_ticks(*from_date*, *to_date*, *date*, ***query_params*)

Returns
Get historic ticks for a forex currency pair.

Parameters

- **from_date** (*required, string, date ex. '2021-07-22'*) – The start of the aggregate time window.
- **to** (*required, string, date ex. '2021-07-22'*) – The end of the aggregate time window.
- **date** (*required, string, date ex. '2021-07-22'*) – The beginning date for the aggregate window.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **offset** (*optional, string, date or nanosecond timestamp*) – The timestamp offset, used for pagination. This is the offset at which to start the results.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.

get_last_pair_quote(*from_date*, *to_date*, ***query_params*)

Returns
Get the last quote tick for a forex currency pair.

Parameters

- **from_date** (*required, string, date ex. '2021-07-22'*) – The start of the aggregate time window.
- **to** (*required, string, date ex. '2021-07-22'*) – The end of the aggregate time window.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.

get_previous_close(***query_params*)

Returns
Get the previous day's open, high, low, and close (OHLC) for the specified forex pair.

Parameters

- **forex_ticker** (*required, string*) – The ticker symbol of the currency pair.

- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **adjusted** (*optional, boolean, default True*) – Whether or not the results are adjusted for splits. Set this to false to get results that are NOT adjusted for splits.

get_quotes(***query_params*)

Returns
Get BBO quotes for a ticker symbol in a given time range.

Parameters

- **forex_ticker** (*required, string*) – The ticker symbol of the currency pair.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **timestamp** (*optional, string, date or nanosecond timestamp*) – Query by timestamp. Either a date with the format YYYY-MM-DD or a nanosecond timestamp.
- **order** (*optional, string, [desc, asc]*) – Order results based on the sort field.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.
- **sort** (*optional, string*) – Sort field used for ordering.

get_snapshot_all(***query_params*)

Returns
Get the current minute, day, and previous day's aggregate, as well as the last trade and quote for all traded forex symbols.

Parameters

- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **tickers** (*optional, string, list of tickers*) – A comma separated list of tickers to get snapshots for.

get_snapshot_gain_lose(*direction*, ***query_params*)

Returns
Get the current top 20 gainers or losers of the day in forex markets.

Parameters

- **direction** (*required, string, [gainers, losers]*) – The direction of the snapshot results to return.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.

get_snapshot_ticker(***query_params*)

:return:Get the current minute, day, and previous day's aggregate, as well as the last trade and quote for a single traded currency symbol.

Parameters

- **forex_ticker** (*required, string*) – The ticker symbol of the currency pair.
- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.

6.4 polygonReference Class

class polygon.**polygonReference**(*external=None*)

get_conditions(***query_params*)

Returns
List all conditions that Polygon.io uses.

Parameters

- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **asset_class** (*optional, string*) – The asset class by which you would like to filter results.
- **data_type** (*optional, string*) – The data type by which you would like to filter results.
- **id** (*optional, integer*) – Filter for conditions with a given ID.
- **sip** (*optional, string, [CTA, UTP, OPRA]*) – Filter by SIP. If the condition contains a mapping for that SIP, the condition will be returned.
- **order** (*optional, string, [desc, asc]*) – Order results based on the sort field.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.
- **sort** (*optional, string*) – Sort field used for ordering.

get_exchanges(***query_params*)

Returns
List all exchanges that Polygon.io knows about.

Parameters

- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **asset_class** (*optional, string*) – The asset class by which you would like to filter results.
- **locale** (*optional, string*) – The locale by which you would like to filter results.

get_market_holidays(***query_params*)

Returns
Get upcoming market holidays and their open/close times.

Parameters
external (*optional, string*) – Your Polygon API key if you are not using environment variables.

get_market_status(***query_params*)

Returns
Get the current trading status of the exchanges and overall financial markets.

Parameters
external (*optional, string*) – Your Polygon API key if you are not using environment variables.

get_ticker_news(***query_params*)

Returns
Get the most recent news articles relating to a stock ticker symbol, including a summary of the article and a link to the original source.

Parameters

- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **ticker** (*optional, string*) – The ticker for which you would like to query information for.
- **published_utc** (*optional, string, date ex. '2021-07-22'*) – Return results published on, before, or after this date.
- **order** (*optional, string, [desc, asc]*) – Order results based on the sort field.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.
- **sort** (*optional, string*) – Sort field used for ordering.

get_ticker_types(***query_params*)

Returns
List all ticker types that Polygon.io has.

Parameters

- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **asset_class** (*optional, string*) – The asset class by which you would like to filter results.
- **locale** (*optional, string*) – The locale by which you would like to filter results.

get_tickers(***query_params*)

Returns
Query all ticker symbols which are supported by Polygon.io.

Parameters

- **external** (*optional, string*) – Your Polygon API key if you are not using environment variables.
- **ticker** (*optional, string*) – The ticker for which you would like to query information for.
- **type** (*optional, string, default is to query all types*) – The type of the ticker, Available through Polygon.io's Ticker Types API.
- **market** (*optional, string, default is to query all markets*) – The market you would like to filter results by.
- **exchange** (*optional, string, default is to query all exchanges*) – The primary exchange of the asset in the ISO code format.
- **cusip** (*optional, string, default is to query all CUSIPs*) – The CUSIP code of the asset you want to search for.

- **cik** (*optional, string, default is to query all CIKs*) – The CIK of the asset you want to search for.
- **date** (*optional, string, date ex. '2021-07-22', default is most recent date*) – Specify a point in time to retrieve tickers available on that date.
- **search** (*optional, string*) – Search for terms within the ticker and/or company name.
- **active** (*optional, boolean, default is true*) – Specify if the tickers returned should be actively traded on the queried date
- **sort** (*optional, string*) – Sort field used for ordering.
- **order** (*optional, string, [desc, asc]*) – Order results based on the sort field.
- **limit** (*optional, integer, default 5000, max 50000*) – Limits the number of base aggregates queried to create the aggregate results.

CHAPTER

SEVEN

US GOVERNMENT ECONOMIC SERIES DATA FROM FRED API

CHAPTER

EIGHT

US GOVERNMENT ELECTRONIC DATA GATHERING, ANALYSIS, AND RETRIEVAL SYSTEM (EDGAR)

class edgar.**edgarFiler**(*ticker*, *name='webmaster'*, *email='webmaster@sec.gov'*)

Accesses data from the SEC's EDGAR database RESTful API. In current state this class wraps all the functionality for the API, providing JSON data to the user. The SEC asks that users do not make more than 10 requests per second. The SEC requires that automated requests declare a user-agent header, and while this class provides a default it is highly recommended that you set your own name and email.

Parameters

- **ticker** (*string, required*) – The ticker symbol of the stock which you would like to access data for.
- **name** (*string, strongly advised*) – Your personal or business name. Used in request headers to validate your API access.
- **email** (*string, strongly advised.*) – Your personal or business email. Used in request headers to validate your API access.

get_accessions(*count=20*, *filter=None*)

Returns most recent accession numbers and document type for the document for the desired company If there is no filter set, it sets and attribute called *accession_numbers* with result. If there are filters applied, the filters are joined with underscores in this format: *{filters}_accessions*.

Data comes from the *data.sec.gov/submissions/* endpoint. The SEC asks that users do not make more than 10 requests per second.

Parameters

- **count** (*int, optional, default 20*) – The number of documents to return.
- **filter** (*list of strings, optional, defualt is None*) – Filters results by the requested form type. Accepts multiple values.

Returns

dictionary, {accession_number : {form : filing_type,date : filing_date}}

get_cik()

Finds the Central Index Key (CIK) for the requested ticker through HTML tree traversal. :return: string, CIK

get_company_concept(*concept*, *taxonomy='us-gaap'*)

Returns all the disclosures from a single company and concept (a taxonomy and tag) into a single JSON file. Returns a separate array of facts for each units on measure that the company has chosen to disclose. Sets an attribute for the class formatted like *{concept}_{taxonomy}* which allows you to fetch multiple taxonomies of the same concept.

Data comes from the *data.sec.gov/api/xbrl/companyconcept/* endpoint. The SEC asks that users do not make more than 10 requests per second.

Parameters

- **concept** (*string, required*) – The tag or line item you are requesting.
- **taxonomy** (*string, optional, default is us-gaap*) – The reporting taxonomy for the tag you want to access. (us-gaap, ifrs-full, dei, srt)

Returns
Dictionary of raw JSON data returned by endpoint

get_company_facts()

Returns all the company concepts data for a company into a single JSON object. Sets a class attribute called *company_facts*.

Data comes from the *data.sec.gov/api/xbrl/companyfacts/* endpoint. The SEC asks that users do not make more than 10 requests per second.

Returns
Dictionary of raw JSON returned by the endpoint

get_edgar_request(*url*, *user_header=True*, *stream=False*)

Returns a response object from EDGAR with options for headers and file streaming. Throws a verbose error code on non-200 status codes. Saves all raw responses in *request_log* class attribute.

Parameters

- **url** (*string, required*) – The url you would like to request.
- **user_header** – If you would like to send

get_report_raw(*accession*, *directory=''*)

Takes an accession number and streams a text report file to your environment. File is named in the following format: *{ticker}_{accession}*. You can use the directory argument to specify the directory you would like to save the file in. This uses the exact text entered and appends it to the filename allowing for relative file placement.

Parameters

- **accession** (*string, required*) – The accession number for the report you would like to download. Available online or through *get_accessions method*.
- **directory** (*string, optional*) – The directory you would like to send the file to.

Returns
filename, string

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

i

p

INDEX

H

I

K

L

M

N

O

P

Q

R

S

T

V