Virtual Finance API Documentation

Release 0.6.0

Feite Brekeveld

Jun 01, 2021

Contents:

CHAPTER 1

Virtual Finance API

1.1 Interactive

Using the Jupyter notebook it is easy to experiment with the *virtual_finance_api* library.

1.2 Install

```
# Setup a virtual environment
$ mkdir vfa
$ cd vfa
$ python3 -m venv venv38
$ . ./venv38/bin/activate
(venv38) feite@salmay:~/vfa$

$ pip install virtual_finance_api
```

1.2.1 . . . Or the latest

With *virtual_finance_api* installed, it is directly available via the commandline:

```
$ virtual_finance_api --help
Usage: vfapi [OPTIONS] COMMAND [ARGS]...

  Virtual Finance API commandline app.

Options:
  --help  Show this message and exit.

Commands:
  financials
  history
  holders
  profile     Profile.
```

. . . additional help on the *history* command:

```
vfapi history --help
Usage: vfapi history [OPTIONS] TICKER

Options:
  --period [1d|5d|1mo|3mo|6mo|1y|2y|5y|10y|ytd|max]
  --interval [1m|2m|5m|15m|30m|60m|90m|1h|1d|5d|1wk|1mo|3mo]
  --csv
  --help                          Show this message and exit.
```

So, lets query for some history for IBM . . .

```
$ vfapi history IBM
                            Open        High         Low       Close   Volume
2021-03-01 14:30:00  120.349998  122.320000  119.860001  120.739998  5714500
2021-03-02 14:30:00  120.739998  121.900002  120.260002  120.330002  4522200
2021-03-03 14:30:00  120.500000  122.629997  119.980003  122.360001  7396200
2021-03-04 14:30:00  122.000000  123.220001  118.760002  120.110001  8062100
2021-03-05 14:30:00  120.639999  123.750000  120.250000  122.830002  6944900
2021-03-08 14:30:00  122.989998  126.849998  122.879997  124.809998  7236600
2021-03-09 14:30:00  125.400002  126.430000  124.160004  124.180000  5608200
2021-03-10 14:30:00  125.050003  128.240005  124.610001  127.870003  7243500
2021-03-11 14:30:00  128.089996  128.639999  126.779999  127.139999  5145000
2021-03-12 14:30:00  127.190002  127.680000  126.610001  127.610001  4009600
2021-03-15 13:30:00  127.769997  128.750000  127.540001  128.580002  3420600
2021-03-16 13:30:00  128.279999  128.520004  127.339996  128.240005  4630400
2021-03-17 13:30:00  128.460007  129.490005  127.489998  129.029999  4244800
2021-03-18 13:30:00  128.940002  131.000000  127.790001  130.059998  5834600
2021-03-19 13:30:00  130.020004  130.440002  128.529999  128.899994  9830600
2021-03-22 13:30:00  128.500000  130.720001  127.889999  130.550003  4164900
2021-03-23 13:30:00  130.440002  131.559998  129.800003  130.460007  4356400
2021-03-24 13:30:00  130.949997  132.110001  130.570007  130.619995  4005000
2021-03-25 13:30:00  130.330002  133.240005  129.770004  133.070007  5554000
2021-03-26 13:30:00  133.289993  136.479996  133.119995  136.380005  5562500
2021-03-29 13:30:00  135.979996  137.070007  135.509995  135.860001  4620900
```

The *Virtual Finance API* provides access to data from financial sites as if it is accessing a REST-API. Currently covered:

- yahoo 'endpoints' to get:
  - financials
  - history

  - holders
  - options
  - profile
  - screener
  - screeners
  - yahooindex
- business inisder 'endpoint':
  - fetch ISIN code
- yfinance compatibility 'endpoints'
- standardized JSON 'endpoints'

With *request-classes* for these endpoints, getting data is as easy as:

```
>>> import json
>>> import virtual_finance_api as fa
>>> import virtual_finance_api.endpoints.yahoo as yh

>>> client = fa.Client()
>>> r = yh.Holders('IBM')
>>> rv = client.request(r)
# lets get the 'major' holders from that JSON response
>>> print(json.dumps(rv['major'], indent=2))

{
   "0": {
     "0": "0.13%",
     "1": "58.58%",
     "2": "58.66%",
     "3": "2561"
   },
   "1": {
     "0": "% of Shares Held by All Insider",
     "1": "% of Shares Held by Institutions",
     "2": "% of Float Held by Institutions",
     "3": "Number of Institutions Holding Shares"
   }
}
```

With the *extensions.stdjson* endpoints this looks like:

```
>>> import virtual_finance_api.extensions.stdjson.endpoints as je
>>> client = fa.Client()
>>> r = je.Holders('IBM')
>>> rv = client.request(r)
# lets get the 'major' holders from that JSON response
>>> print(json.dumps(rv['major'], indent=2))

[
  [
    "0.13%",
    "% of Shares Held by All Insider"
  ],
  [
```

(continues on next page)

(continued from previous page)

```
    "58.25%",
    "% of Shares Held by Institutions"
  ],
  [
    "58.33%",
    "% of Float Held by Institutions"
  ],
  [
    "2696",
    "Number of Institutions Holding Shares"
  ]
]
```

It make more sense to group the information. The base classes simply pass the JSON data the way it is scraped from the source URL. Derived classes can be used to transform this data. The *extension.stdjson* performs this task.

1.3 Yfinance compatibility

There is a compatibility layer with Yfinance too. It provides requests derived from the base requests, extended with properties that give the same information as Yfinance does.

The *Holders*-example from above becomes:

```
>>> import json
>>> import virtual_finance_api as fa
>>> import virtual_finance_api.compat.yfinance.endpoints as yf

>>> client = fa.Client()
>>> r = yf.Holders('IBM')
>>> rv = client.request(r)
>>> # lets get the 'major' holders from that JSON response
>>> print(r.major)


        0                                         1
0   0.13%        % of Shares Held by All Insider
1  58.58%       % of Shares Held by Institutions
2  58.66%        % of Float Held by Institutions
3    2561  Number of Institutions Holding Shares

>>> # or, that same information from the dataframe in JSON
>>> # (dump, load, dump to 'pretty print')
>>> print(json.dumps(json.loads(r.major.to_json()), indent=2))
{
   "0": {
     "0": "0.13%",
     "1": "58.58%",
     "2": "58.66%",
     "3": "2561"
   },
   "1": {
     "0": "% of Shares Held by All Insider",
     "1": "% of Shares Held by Institutions",
     "2": "% of Float Held by Institutions",
     "3": "Number of Institutions Holding Shares"
```

(continues on next page)

(continued from previous page)

```
    }
}

>>> print(r.institutional)
                                      Holder     Shares Date Reported   % Out
↪Value
0                  Vanguard Group, Inc. (The)   73806391    2020-12-30  0.0826
↪9290748499
1                             Blackrock Inc.   62271273    2020-12-30  0.0697
↪7838707845
2                       State Street Corporation  51941856    2020-12-30  0.0581
↪6538440833
3                Geode Capital Management, LLC  13310817    2020-12-30  0.0149
↪1675565643
4  Charles Schwab Investment Management, Inc.  12571878    2020-12-30  0.0141
↪1582548002
5                   Northern Trust Corporation  10652880    2020-12-30  0.0119
↪1340984534
6                             Morgan Stanley   9853901    2020-12-30  0.0110
↪1240409057
7          Bank Of New York Mellon Corporation   9628160    2020-12-30  0.0108
↪1211992780
8            Norges Bank Investment Management   8865649    2020-12-30  0.0099
↪1116007896
9                  Bank of America Corporation   8074146    2020-12-30  0.0090
↪1016373498
```

See the https://virtual-finance-api.readthedocs.io/en/latest/?badge=latest for details.

1.4 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

1.4.1 Installation

Stable release

To install Virtual Finance API, run this command in your terminal:

```
$ pip install virtual_finance_api
```

This is the preferred method to install Virtual Finance API, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

From sources

The sources for Virtual Finance API can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/hootnot/virtual-finance-API
```

Or download the tarball:

```
$ curl -OJL https://github.com/hootnot/virtual-finance-API/tarball/main
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

1.4.2 Motivation

While building a service on a task based concept using Celery or Dramatiq, I ran into connection related issues using for instance a library like yfinance . These issues occured when firing a large number of requests asynchronously.

In the need of an, over all, more fine grained control I, decided to write a new library:

- easy to scale
- easy to maintain
- easy to extend
- integrated logging
- compatibility layers
  - *yfinance* compatibility package

REST-API

Since I'm a fan of REST-API's, the thought was to approach this as a wrapper for a REST-API, in this case a *"virtual-REST-API"*.

Two basic components

> *Request* classes to represent the specfic requests providing an API like a REST-API with *route-parameters* and *query parameters*. When executed the request instance will hold the response and the HTTP-status.
>
> A *Client* performing the requests.

Since finance.yahoo.com provides no real REST-API, the requests of the wrappper fake the REST-API. A *request* instance holds the details for the URL to call. In case it is HTML-data that needs to be processed to scrape the information, a *conversion* method is run after the HTML-data is received.

This second stage of processing gives the desired JSON-response or, in case of errors some 4xx error code is returned. The client processes this the same way as if it was the response of a REST-API.

In case the second stage processing fails the original status code is set to a code identifying the final state:

- if processing was without errors, but the desired result could not be created, the status code is changed to 404, 'Not Found'
- in case processing gave errors, the status code is changed to 422, 'Unprocessable Entity'

These status codes represent a REST-API server responding to client requests.

There are a handful of endpoints that provide the information we need. The request classes wrap the logic to provide the JSON data fetched (or fabricated) from the response, acting as if that response was provided by site that was called.

1.4.3 Usage

To use Virtual Finance API in a project simply create a *Client* and *Requests* to be performed by the *Client* like:

```
import virtual_finance_api as fa
import virtual_finance_api.endpoints.yahoo as yh
import json


client = fa.Client()
ticker = 'IBM'
r = yh.Profile(ticker)
print(json.dumps(client.request(r), indent=2))

 {
   "defaultKeyStatistics": {
     "annualHoldingsTurnover": null,
     "enterpriseToRevenue": 2.274,
     "beta3Year": null,
     "profitMargins": 0.07593,
     "enterpriseToEbitda": 10.955,
     "52WeekChange": 0.2859279,
     "morningStarRiskRating": null,
     "forwardEps": 12.08,
     "revenueQuarterlyGrowth": null,
     "sharesOutstanding": 893593984,
     "fundInceptionDate": null,
     "annualReportExpenseRatio": null,
     "totalAssets": null,
     "bookValue": 23.074,
     "sharesShort": 30005072,
     "sharesPercentSharesOut": 0.0336,
     "fundFamily": null,
     "lastFiscalYearEnd": 1609372800,
     "heldPercentInstitutions": 0.58584,
     "netIncomeToCommon": 5501000192,
     "trailingEps": 6.233,
     ....              And a lot more ....
     "maxAge": 86400
   },
   "pageViews": {
     "shortTermTrend": "UP",
     "midTermTrend": "UP",
     "longTermTrend": "UP",
     "maxAge": 1
   }
 }
```

Exceptions

To catch exceptions, wrap the *client.request()* in a try/catch like:

```
import virtual_finance_api as fa
import virtual_finance_api.endpoints.yahoo as yh
import json
```

(continues on next page)

(continued from previous page)

```
client = fa.Client()
ticker = 'IBM'
r = yh.Profile(ticker)
try:
    rv = client.request(r)

except fa.VirtualFinanceAPIError as err:
    print(err)

else:
    print(json.dumps(rv, indent=2))
```

Logging

The *virtual_finance_library* has logging integrated. By simply importing the *logging* module and configuring it, you will have logging available.

```
import virtual_finance_api as fa
import virtual_finance_api.endpoints.yahoo as yh
import json
import logging


logging.basicConfig(
    filename="./your_appl_name.log",
    level=logging.INFO,
    format='%(asctime)s [%(levelname)s] %(name)s : %(message)s',
)

client = fa.Client()
ticker = 'IBM'
r = yh.Profile(ticker)
try:
    rv = client.request(r)

except fa.VirtualFinanceAPIError as err:
    print(err)

else:
    print(json.dumps(rv, indent=2))
```

... and the log looks like:

```
2021-04-09 14:26:28,884 [INFO] virtual_finance_api.client : performing␣
↪request https://finance.yahoo.com/quote/IBM
2021-04-09 14:26:29,359 [INFO] virtual_finance_api.endpoints.yahoo.ticker_
↪bundle : conversion_hook: IBM OK
```

1.4.4 Reference

virtual_finance_api

Client and Exceptions

Top-level package for Virtual Finance API.

class virtual_finance_api.**Client**(*headers: dict = None*, *request_params: dict = None*)
Bases: object

__init__(*headers: dict = None*, *request_params: dict = None*) → None
Instantiate a Client instance.

Parameters

- **headers** (*dict (optional)*) – optional headers to set to requests
- **request_params** (*dict (optional)*) – optional parameters to set to requests

for details pls. check requests.readthedocs.io

Example

```
>>> import virtual_finance_api as fa
>>> import virtual_finance_api.endpoints.yahoo as yh
>>> import json
>>> client = fa.Client()
>>> r = yh.Profile('IBM')
>>> try:
...     rv = client.request(r)
... except VirtualFinanceAPIError as err:
...     print(err)
... else:
...     print(json.dumps(rv['pageViews'], indent=2))
{
  "shortTermTrend": "NEUTRAL",
  "midTermTrend": "UP",
  "longTermTrend": "UP",
  "maxAge": 1
}
```

request(*endpoint: virtual_finance_api.endpoints.apirequest.VirtualAPIRequest*)
Perform a request for the APIRequest instance 'endpoint'.

Parameters **endpoint** (*APIRequest (required)*) – The endpoint parameter contains an instance of an APIRequest containing the endpoint, method and optionally other parameters or body data.

Raises

- ***VirtualFinanceAPIError*** – in case of HTTP response code >= 400
- **requests-lib exceptions** – Possible exceptions from the requests library, those are re-raised.

request_params
request_params property.

exception virtual_finance_api.**VirtualFinanceAPIError**(*code: int*, *msg: str*)
Bases: Exception

Generic error class. In case of HTTP response codes >= 400 this class can be used to raise an exception representing that error.

__init__(*code: int*, *msg: str*)
Instantiate a VirtualFinanceError.

Parameters

- **code** (*int*) – the HTTP-code of the response
- **msg** (*str*) – the message returned with the response

Yahoo endpoints

class virtual_finance_api.endpoints.yahoo.**Profile**(*ticker: str*)
Bases: virtual_finance_api.endpoints.apirequest.VirtualAPIRequest, virtual_finance_api.endpoints.yahoo.ticker_bundle.Yhoo

Profile - class to handle the profile endpoint.

DOMAIN = 'https://finance.yahoo.com'

ENDPOINT = 'quote/{ticker}'

EXPECTED_STATUS = 200

METHOD = 'GET'

RESPONSE_TYPE = 'txt'

__init__(*ticker: str*) → None
Instantiate a Profile APIRequest instance.

Parameters **ticker** (*string (required)*) – the ticker to perform the request for.

Example

```
>>> import virtual_finance_api as fa
>>> import virtual_finance_api.endpoints.yahoo as yh
>>> client = fa.Client()
>>> r = yh.Profile('IBM')
>>> rv = client.request(r)
>>> print(json.dumps(rv, indent=2))
```

```
{
  "info": {
    "zip": "10504",
    "sector": "Technology",
    "fullTimeEmployees": 345900,
    "longBusinessSummary": "International Business Machines Corporation ......
↪.",
    "city": "Armonk",
    "phone": "914 499 1900",
    "state": "NY",
    "country": "United States",
    "companyOfficers": [],
    "website": "http://www.ibm.com",
    "maxAge": 1,
    "address1": "One New Orchard Road",
    "industry": "Information Technology Services",
    "previousClose": 131.18,
```

(continues on next page)

(continued from previous page)


```
    "regularMarketOpen": 131.305,
    "twoHundredDayAverage": 123.6211,
    "trailingAnnualDividendYield": 0.049626473,
    "payoutRatio": 1.0619999,
    "volume24Hr": null,
    "regularMarketDayHigh": 132.66,
    "navPrice": null,
    "averageDailyVolume10Day": 4423514,
    "totalAssets": null,
    "regularMarketPreviousClose": 131.18,
    "fiftyDayAverage": 128.79886,
    "trailingAnnualDividendRate": 6.51,
    "open": 131.305,
    "toCurrency": null,
    "averageVolume10days": 4423514,
    "expireDate": null,
    "yield": null,
    "algorithm": null,
    "dividendRate": 6.52,
    "exDividendDate": 1612828800,
    "beta": 1.225041,
    "circulatingSupply": null,
    "...": "..."
  }
}
```


class virtual_finance_api.endpoints.yahoo.**History**(*ticker: str*, *params: dict*)

Bases: virtual_finance_api.endpoints.apirequest.VirtualAPIRequest, virtual_finance_api.endpoints.yahoo.ticker_bundle.Yhoo

History - class to handle the history endpoint.

DOMAIN = 'https://query1.finance.yahoo.com'

ENDPOINT = 'v8/finance/chart/{ticker}'

EXPECTED_STATUS = 200

METHOD = 'GET'

RESPONSE_TYPE = 'json'

__init__(*ticker: str*, *params: dict*) → None

Instantiate a History APIRequest instance.

Parameters

- **ticker** (*string (required)*) – the ticker to perform the request for.
- **params** (*dict (optional)*) – dictionary with optional parameters to perform the request, parameters default to 1 month of daily (1d) historical data.


```
{
  "interval": "1d",
  "period": "max",
  "actions": true
}
```

```
>>> import virtual_finance_api as fa
>>> import virtual_finance_api.endpoints.yahoo as yh
>>> client = fa.Client()
>>> r = yh.History('IBM', params=params)
>>> rv = client.request(r)
```

```
>>> print(r.response)
```


```
{
  "meta": {
    "currency": "USD",
    "symbol": "IBM",
    "exchangeName": "NYQ",
    "instrumentType": "EQUITY",
    "firstTradeDate": -252322200,
    "regularMarketTime": 1618419277,
    "gmtoffset": -14400,
    "timezone": "EDT",
    "exchangeTimezoneName": "America/New_York",
    "regularMarketPrice": 132.23,
    "chartPreviousClose": 127.61,
    "priceHint": 2,
    "currentTradingPeriod": {
      "pre": {
        "timezone": "EDT",
        "start": 1618387200,
        "end": 1618407000,
        "gmtoffset": -14400
      },
      "regular": {
        "timezone": "EDT",
        "start": 1618407000,
        "end": 1618430400,
        "gmtoffset": -14400
      },
      "post": {
        "timezone": "EDT",
        "start": 1618430400,
        "end": 1618444800,
        "gmtoffset": -14400
      }
    },
    "dataGranularity": "1d",
    "range": "1mo",
    "validRanges": [
      "1d",
      "5d",
      "1mo",
      "3mo",
      "6mo",
      "1y",
      "2y",
      "5y",
      "10y",
      "ytd",
      "max"
    ]
```


(continues on next page)

(continued from previous page)


```
  },
  "ohlcdata": {
    "timestamp": [
      1615815000,
      1615901400,
      1615987800,
      1616074200,
      1616160600,
      1616419800,
      "..."
    ],
    "open": [
      "..."
    ]
  }
}
```


class virtual_finance_api.endpoints.yahoo.**Holders**(*ticker: str*)

Bases: virtual_finance_api.endpoints.apirequest.VirtualAPIRequest, virtual_finance_api.endpoints.yahoo.ticker_bundle.Yhoo

Holders - class to handle the holders endpoint.

DOMAIN = 'https://finance.yahoo.com'

ENDPOINT = 'quote/{ticker}/holders'

EXPECTED_STATUS = 200

METHOD = 'GET'

RESPONSE_TYPE = 'txt'

__init__(*ticker: str*) → None

Instantiate a Holders APIRequest instance.

Parameters **ticker** (*string (required)*) – the ticker to perform the request for.

Example

```
>>> import virtual_finance_api as fa
>>> import virtual_finance_api.endpoints.yahoo as yh
>>> client = fa.Client()
>>> r = yh.Holders('IBM')
>>> rv = client.request(r)
>>> print(json.dumps(rv, indent=2))
```


```
{
  "major": [
    [
      "0.13%",
      "% of Shares Held by All Insider"
    ],
    [
      "58.24%",
      "% of Shares Held by Institutions"
    ],
```


(continues on next page)

(continued from previous page)


```
    [
      "58.32%",
      "% of Float Held by Institutions"
    ],
    [
      "2692",
      "Number of Institutions Holding Shares"
    ]
  ],
  "institutional": {
    "legend": {
      "holder": "Holder",
      "shares": "Shares",
      "date_reported": "Date Reported",
      "pch_out": "% Out",
      "value": "Value"
    },
    "holders": [
      {
        "holder": "Vanguard Group, Inc. (The)",
        "shares": 73806391,
        "date_reported": "Dec 30, 2020",
        "pch_out": 8.26,
        "value": 9290748499
      },
      {
        "holder": "Blackrock Inc.",
        "shares": 62271273,
        "date_reported": "Dec 30, 2020",
        "pch_out": 6.97,
        "value": 7838707845
      },
      {
        "...": "..."
      }
    ]
  }
}
```


class virtual_finance_api.endpoints.yahoo.**Financials**(*ticker: str*)

Bases: virtual_finance_api.endpoints.apirequest.VirtualAPIRequest, virtual_finance_api.endpoints.yahoo.ticker_bundle.Yhoo

Financials - class to handle the financials endpoint.

DOMAIN = 'https://finance.yahoo.com'

ENDPOINT = 'quote/{ticker}/financials'

EXPECTED_STATUS = 200

METHOD = 'GET'

RESPONSE_TYPE = 'txt'

__init__(*ticker: str*) → None

Instantiate a Financials APIRequest instance.

Parameters **ticker** (*string (required)*) – the ticker to perform the request for.

Example

```
>>> import virtual_finance_api as fa
>>> import virtual_finance_api.endpoints.yahoo as yh
>>> client = fa.Client()
>>> r = yh.Financials('IBM')
>>> rv = client.request(r)
>>> print(json.dumps(rv, indent=2))
```


```
{
  "cashflow": {
    "yearly": [
      {
        "investments": -628000000,
        "changeToLiabilities": 138000000,
        "totalCashflowsFromInvestingActivities": -3028000000,
        "netBorrowings": -3714000000,
        "totalCashFromFinancingActivities": -9721000000,
        "changeToOperatingActivities": 3023000000,
        "netIncome": 5590000000,
        "changeInCash": 5361000000,
        "endDate": 1609372800,
        "repurchaseOfStock": -302000000,
        "effectOfExchangeRate": -87000000,
        "totalCashFromOperatingActivities": 18197000000,
        "depreciation": 6695000000,
        "dividendsPaid": -5797000000,
        "changeToInventory": -209000000,
        "changeToAccountReceivables": 5297000000,
        "otherCashflowsFromFinancingActivities": 92000000,
        "maxAge": 1,
        "changeToNetincome": -2337000000,
        "capitalExpenditures": -2618000000
      },
      {
        "investments": 268000000,
        "changeToLiabilities": -503000000,
        "totalCashflowsFromInvestingActivities": -26936000000,
        "netBorrowings": 16284000000,
        "totalCashFromFinancingActivities": 9042000000,
        "changeToOperatingActivities": 1159000000,
        "netIncome": 9431000000,
        "changeInCash": -3290000000,
        "endDate": 1577750400,
        "repurchaseOfStock": -1633000000,
        "effectOfExchangeRate": -167000000,
        "totalCashFromOperatingActivities": 14770000000,
        "depreciation": 6059000000,
        "dividendsPaid": -5707000000,
        "changeToInventory": 67000000,
        "changeToAccountReceivables": 502000000,
        "otherCashflowsFromFinancingActivities": 98000000,
        "maxAge": 1,
        "changeToNetincome": -1945000000,
        "capitalExpenditures": -2286000000
      },
      {
```


(continues on next page)

(continued from previous page)

```
        "...": "... etc. ..."
      }
    ]
  }
}
```

class virtual_finance_api.endpoints.yahoo.**Options**(*ticker: str*, *params: dict = None*)

Bases: virtual_finance_api.endpoints.apirequest.VirtualAPIRequest, virtual_finance_api.endpoints.yahoo.ticker_bundle.Yhoo

Options - class to handle the options endpoint.

DOMAIN = 'https://query1.finance.yahoo.com'

ENDPOINT = 'v7/finance/options/{ticker}'

EXPECTED_STATUS = 200

METHOD = 'GET'

RESPONSE_TYPE = 'json'

__init__(*ticker: str*, *params: dict = None*) → None

Instantiate a Options APIRequest instance.

Parameters

- **ticker** (*string (required)*) – the ticker to perform the request for.
- **params** – dict with optional 'date' parameter.

Example

```
>>> import virtual_finance_api as fa
>>> import virtual_finance_api.endpoints.yahoo as yh
>>> client = fa.Client()
>>> r = yh.Options('IBM')
>>> rv = client.request(r)
>>> print(json.dumps(rv, indent=2))
```

```
{
  "underlyingSymbol": "IBM",
  "expirationDates": [
    1618531200,
    1619136000,
    1619740800,
    1620345600,
    1620950400,
    1621555200,
    1622160000,
    1623974400,
    1626393600,
    1631836800,
    1634256000,
    1642723200,
    1655424000,
    1674172800
  ],
```

(continues on next page)

(continued from previous page)


```
  "strikes": [
    55.0,
    60.0,
    65.0,
    70.0,
    75.0,
    80.0,
    85.0,
    90.0,
    95.0,
    "...",
    165.0,
    170.0,
    175.0,
    180.0,
    185.0,
    190.0,
    195.0,
    200.0
  ],
  "hasMiniOptions": false,
  "options": [
    {
      "expirationDate": 1618531200,
      "hasMiniOptions": false,
      "calls": [
        {
          "contractSymbol": "IBM210416C00060000",
          "strike": 60.0,
          "currency": "USD",
          "lastPrice": 67.05,
          "change": 0.0,
          "percentChange": 0.0,
          "volume": 1,
          "openInterest": 2,
          "bid": 73.2,
          "ask": 76.65,
          "contractSize": "REGULAR",
          "expiration": 1618531200,
          "lastTradeDate": 1615476051,
          "impliedVolatility": 6.591798635253907,
          "inTheMoney": true
        },
        {
          "...": " ... etc ..."
        }
      ]
    }
  ]
}
```


class virtual_finance_api.endpoints.yahoo.**Screener**(*name: str*)

Bases: virtual_finance_api.endpoints.apirequest.VirtualAPIRequest

Screener - class to handle the screener endpoint.

DOMAIN = 'https://finance.yahoo.com'

ENDPOINT = 'screener/predefined/{name}'

EXPECTED_STATUS = 200

METHOD = 'GET'

RESPONSE_TYPE = 'txt'

__init__(*name: str*) → None
Instantiate a Screener request instance.

Parameters **name** (*string (required)*) – the name of the predefined screener to perform the request for.

```
>>> import virtual_finance_api as fa
>>> import virtual_finance_api.endpoints.yahoo as yh
>>> client = fa.Client()
>>> r = yh.Screener('MOST_SHORTED_STOCKS')
>>> rv = client.request(r)
```


```
{
  "rows": [
    {
      "symbol": "ESPR",
      "shortName": "Esperion Therapeutics, Inc.",
      "longName": "Esperion Therapeutics, Inc.",
      "quoteType": "EQUITY",
      "currency": "USD",
      "regularMarketPrice": {
        "raw": 29.39,
        "fmt": "29.39"
      },
      "regularMarketChange": {
        "raw": 1.0599995,
        "fmt": "1.06"
      },
      "regularMarketChangePercent": {
        "raw": 3.7416148,
        "fmt": "3.74%"
      },
      "regularMarketVolume": {
        "raw": 1268153,
        "fmt": "1.268M",
        "longFmt": "1,268,153"
      },
      "averageDailyVolume3Month": {
        "raw": 842390,
        "fmt": "842,390",
        "longFmt": "842,390"
      },
      "marketCap": {
        "raw": 821233024,
        "fmt": "821.233M",
        "longFmt": "821,233,024"
      },
      "fiftyTwoWeekLow": {
        "raw": 23.9,
        "fmt": "23.90"
      },
```


(continues on next page)

(continued from previous page)


```
      "fiftyTwoWeekHigh": {
        "raw": 53.73,
        "fmt": "53.73"
      },
      "regularMarketOpen": {
        "raw": 28.21,
        "fmt": "28.21"
      },
      "priceHint": 2
    },
    {
      "symbol": "CLVS",
      "shortName": "Clovis Oncology, Inc.",
      "longName": "Clovis Oncology, Inc.",
      "quoteType": "EQUITY",
      "currency": "USD",
      "regularMarketPrice": {
        "raw": 7.86,
        "fmt": "7.86"
      },
      "regularMarketChange": {
        "raw": 2.54,
        "fmt": "2.54"
      },
      "regularMarketChangePercent": {
        "raw": 47.744358,
        "fmt": "47.74%"
      },
      "regularMarketVolume": {
        "raw": 331763706,
        "fmt": "331.764M",
        "longFmt": "331,763,706"
      },
      "averageDailyVolume3Month": {
        "raw": 13614618,
        "fmt": "13.615M",
        "longFmt": "13,614,618"
      },
      "marketCap": {
        "raw": 821605824,
        "fmt": "821.606M",
        "longFmt": "821,605,824"
      },
      "fiftyTwoWeekLow": {
        "raw": 3.98,
        "fmt": "3.98"
      },
      "fiftyTwoWeekHigh": {
        "raw": 11.1,
        "fmt": "11.10"
      },
      "regularMarketOpen": {
        "raw": 6.4,
        "fmt": "6.40"
      },
      "priceHint": 2
    },
```


(continues on next page)

(continued from previous page)


```
    {
      "symbol": "BLNK",
      "shortName": "Blink Charging Co. Common Stock",
      "longName": "Blink Charging Co.",
      "quoteType": "EQUITY",
      "currency": "USD",
      "regularMarketPrice": {
        "raw": 38.24,
        "fmt": "38.24"
      },
      "regularMarketChange": {
        "raw": 2.290001,
        "fmt": "2.29"
      },
      "regularMarketChangePercent": {
        "raw": 6.3699613,
        "fmt": "6.37%"
      },
      "regularMarketVolume": {
        "raw": 5620010,
        "fmt": "5.62M",
        "longFmt": "5,620,010"
      },
      "averageDailyVolume3Month": {
        "raw": 11127281,
        "fmt": "11.127M",
        "longFmt": "11,127,281"
      },
      "marketCap": {
        "raw": 1591166464,
        "fmt": "1.591B",
        "longFmt": "1,591,166,464"
      },
      "fiftyTwoWeekLow": {
        "raw": 1.3,
        "fmt": "1.30"
      },
      "fiftyTwoWeekHigh": {
        "raw": 64.5,
        "fmt": "64.50"
      },
      "regularMarketOpen": {
        "raw": 36.13,
        "fmt": "36.13"
      },
      "priceHint": 2
    },
    {
      "symbol": "SKT",
      "shortName": "Tanger Factory Outlet Centers, ",
      "longName": "Tanger Factory Outlet Centers, Inc.",
      "quoteType": "EQUITY",
      "currency": "USD",
      "regularMarketPrice": {
        "raw": 16.83,
        "fmt": "16.83"
      },
```


(continues on next page)

(continued from previous page)


```
      "regularMarketChange": {
        "raw": -0.3600006,
        "fmt": "-0.36"
      },
      "regularMarketChangePercent": {
        "raw": -2.0942445,
        "fmt": "-2.09%"
      },
      "regularMarketVolume": {
        "raw": 2925256,
        "fmt": "2.925M",
        "longFmt": "2,925,256"
      },
      "averageDailyVolume3Month": {
        "raw": 5614301,
        "fmt": "5.614M",
        "longFmt": "5,614,301"
      },
      "marketCap": {
        "raw": 1574779776,
        "fmt": "1.575B",
        "longFmt": "1,574,779,776"
      },
      "fiftyTwoWeekLow": {
        "raw": 4.05,
        "fmt": "4.05"
      },
      "fiftyTwoWeekHigh": {
        "raw": 22.4,
        "fmt": "22.40"
      },
      "regularMarketOpen": {
        "raw": 17,
        "fmt": "17.00"
      },
      "priceHint": 2
    },
    {
      "symbol": "GEO",
      "shortName": "Geo Group Inc (The) REIT",
      "longName": "The GEO Group, Inc.",
      "quoteType": "EQUITY",
      "currency": "USD",
      "regularMarketPrice": {
        "raw": 8.13,
        "fmt": "8.13"
      },
      "regularMarketChange": {
        "raw": 0.16000032,
        "fmt": "0.16"
      },
      "regularMarketChangePercent": {
        "raw": 2.0075324,
        "fmt": "2.01%"
      },
      "regularMarketVolume": {
        "raw": 7753531,
```


(continues on next page)

(continued from previous page)


```
        "fmt": "7.754M",
        "longFmt": "7,753,531"
      },
      "averageDailyVolume3Month": {
        "raw": 4881275,
        "fmt": "4.881M",
        "longFmt": "4,881,275"
      },
      "marketCap": {
        "raw": 986225920,
        "fmt": "986.226M",
        "longFmt": "986,225,920"
      },
      "trailingPE": {
        "raw": 8.648936,
        "fmt": "8.65"
      },
      "fiftyTwoWeekLow": {
        "raw": 6.7,
        "fmt": "6.70"
      },
      "fiftyTwoWeekHigh": {
        "raw": 15.45,
        "fmt": "15.45"
      },
      "regularMarketOpen": {
        "raw": 8.05,
        "fmt": "8.05"
      },
      "priceHint": 2
    },
    {
      "symbol": "VIVE",
      "shortName": "Viveve Medical, Inc.",
      "longName": "Viveve Medical, Inc.",
      "quoteType": "EQUITY",
      "currency": "USD",
      "regularMarketPrice": {
        "raw": 3.42,
        "fmt": "3.4200"
      },
      "regularMarketChange": {
        "raw": -0.48000002,
        "fmt": "-0.4800"
      },
      "regularMarketChangePercent": {
        "raw": -12.307693,
        "fmt": "-12.31%"
      },
      "regularMarketVolume": {
        "raw": 1346134,
        "fmt": "1.346M",
        "longFmt": "1,346,134"
      },
      "averageDailyVolume3Month": {
        "raw": 2191777,
        "fmt": "2.192M",
```


(continues on next page)

(continued from previous page)


```
        "longFmt": "2,191,777"
      },
      "marketCap": {
        "raw": 35368616,
        "fmt": "35.369M",
        "longFmt": "35,368,616"
      },
      "fiftyTwoWeekLow": {
        "raw": 2.54,
        "fmt": "2.5400"
      },
      "fiftyTwoWeekHigh": {
        "raw": 14.4,
        "fmt": "14.4000"
      },
      "regularMarketOpen": {
        "raw": 3.79,
        "fmt": "3.7900"
      },
      "priceHint": 4
    },
    {
      "symbol": "INO",
      "shortName": "Inovio Pharmaceuticals, Inc.",
      "longName": "Inovio Pharmaceuticals, Inc.",
      "quoteType": "EQUITY",
      "currency": "USD",
      "regularMarketPrice": {
        "raw": 10.11,
        "fmt": "10.11"
      },
      "regularMarketChange": {
        "raw": 0.2699995,
        "fmt": "0.27"
      },
      "regularMarketChangePercent": {
        "raw": 2.7438974,
        "fmt": "2.74%"
      },
      "regularMarketVolume": {
        "raw": 11353412,
        "fmt": "11.353M",
        "longFmt": "11,353,412"
      },
      "averageDailyVolume3Month": {
        "raw": 15485901,
        "fmt": "15.486M",
        "longFmt": "15,485,901"
      },
      "marketCap": {
        "raw": 2099169536,
        "fmt": "2.099B",
        "longFmt": "2,099,169,536"
      },
      "fiftyTwoWeekLow": {
        "raw": 6.53,
        "fmt": "6.53"
```


(continues on next page)

(continued from previous page)


```
    },
    "fiftyTwoWeekHigh": {
      "raw": 33.79,
      "fmt": "33.79"
    },
    "regularMarketOpen": {
      "raw": 10.04,
      "fmt": "10.04"
    },
    "priceHint": 2
  },
  {
    "symbol": "MAC",
    "shortName": "Macerich Company (The)",
    "longName": "The Macerich Company",
    "quoteType": "EQUITY",
    "currency": "USD",
    "regularMarketPrice": {
      "raw": 13,
      "fmt": "13.00"
    },
    "regularMarketChange": {
      "raw": -0.32999992,
      "fmt": "-0.33"
    },
    "regularMarketChangePercent": {
      "raw": -2.4756184,
      "fmt": "-2.48%"
    },
    "regularMarketVolume": {
      "raw": 10771702,
      "fmt": "10.772M",
      "longFmt": "10,771,702"
    },
    "averageDailyVolume3Month": {
      "raw": 8939680,
      "fmt": "8.94M",
      "longFmt": "8,939,680"
    },
    "marketCap": {
      "raw": 2011984000,
      "fmt": "2.012B",
      "longFmt": "2,011,984,000"
    },
    "fiftyTwoWeekLow": {
      "raw": 4.81,
      "fmt": "4.81"
    },
    "fiftyTwoWeekHigh": {
      "raw": 25.99,
      "fmt": "25.99"
    },
    "regularMarketOpen": {
      "raw": 13.29,
      "fmt": "13.29"
    },
    "priceHint": 2
```


(continues on next page)

(continued from previous page)


```
    },
    {
      "symbol": "HTZGQ",
      "shortName": "HERTZ GLOBAL HOLDINGS INC",
      "longName": "Hertz Global Holdings, Inc.",
      "quoteType": "EQUITY",
      "currency": "USD",
      "regularMarketPrice": {
        "raw": 1.17,
        "fmt": "1.1700"
      },
      "regularMarketChange": {
        "raw": 0.07999992,
        "fmt": "0.0800"
      },
      "regularMarketChangePercent": {
        "raw": 7.3394427,
        "fmt": "7.34%"
      },
      "regularMarketVolume": {
        "raw": 3663758,
        "fmt": "3.664M",
        "longFmt": "3,663,758"
      },
      "averageDailyVolume3Month": {
        "raw": 5426501,
        "fmt": "5.427M",
        "longFmt": "5,426,501"
      },
      "marketCap": {
        "raw": 182761008,
        "fmt": "182.761M",
        "longFmt": "182,761,008"
      },
      "fiftyTwoWeekLow": {
        "raw": 0.4,
        "fmt": "0.4000"
      },
      "fiftyTwoWeekHigh": {
        "raw": 9.04,
        "fmt": "9.0400"
      },
      "regularMarketOpen": {
        "raw": 1.14,
        "fmt": "1.1400"
      },
      "priceHint": 4
    },
    {
      "symbol": "WKHS",
      "shortName": "Workhorse Group, Inc.",
      "longName": "Workhorse Group Inc.",
      "quoteType": "EQUITY",
      "currency": "USD",
      "regularMarketPrice": {
        "raw": 16.12,
        "fmt": "16.12"
```


(continues on next page)

(continued from previous page)


```
      },
      "regularMarketChange": {
        "raw": 0.7200012,
        "fmt": "0.72"
      },
      "regularMarketChangePercent": {
        "raw": 4.675333,
        "fmt": "4.68%"
      },
      "regularMarketVolume": {
        "raw": 10435994,
        "fmt": "10.436M",
        "longFmt": "10,435,994"
      },
      "averageDailyVolume3Month": {
        "raw": 21423000,
        "fmt": "21.423M",
        "longFmt": "21,423,000"
      },
      "marketCap": {
        "raw": 1990916864,
        "fmt": "1.991B",
        "longFmt": "1,990,916,864"
      },
      "trailingPE": {
        "raw": 23.028572,
        "fmt": "23.03"
      },
      "fiftyTwoWeekLow": {
        "raw": 1.5,
        "fmt": "1.50"
      },
      "fiftyTwoWeekHigh": {
        "raw": 42.96,
        "fmt": "42.96"
      },
      "regularMarketOpen": {
        "raw": 15.46,
        "fmt": "15.46"
      },
      "priceHint": 2
    },
    {
      "symbol": "KPTI",
      "shortName": "Karyopharm Therapeutics Inc.",
      "longName": "Karyopharm Therapeutics Inc.",
      "quoteType": "EQUITY",
      "currency": "USD",
      "regularMarketPrice": {
        "raw": 12.06,
        "fmt": "12.06"
      },
      "regularMarketChange": {
        "raw": 0.020000458,
        "fmt": "0.02"
      },
      "regularMarketChangePercent": {
```


(continues on next page)

(continued from previous page)


```
        "raw": 0.16611676,
        "fmt": "0.17%"
      },
      "regularMarketVolume": {
        "raw": 2934156,
        "fmt": "2.934M",
        "longFmt": "2,934,156"
      },
      "averageDailyVolume3Month": {
        "raw": 2796244,
        "fmt": "2.796M",
        "longFmt": "2,796,244"
      },
      "marketCap": {
        "raw": 900089728,
        "fmt": "900.09M",
        "longFmt": "900,089,728"
      },
      "fiftyTwoWeekLow": {
        "raw": 10.57,
        "fmt": "10.57"
      },
      "fiftyTwoWeekHigh": {
        "raw": 25.979,
        "fmt": "25.98"
      },
      "regularMarketOpen": {
        "raw": 12.1649,
        "fmt": "12.16"
      },
      "priceHint": 2
    },
    {
      "symbol": "VXRT",
      "shortName": "Vaxart, Inc. - Common Stock",
      "longName": "Vaxart, Inc.",
      "quoteType": "EQUITY",
      "currency": "USD",
      "regularMarketPrice": {
        "raw": 6.88,
        "fmt": "6.88"
      },
      "regularMarketChange": {
        "raw": 0.26000023,
        "fmt": "0.26"
      },
      "regularMarketChangePercent": {
        "raw": 3.927496,
        "fmt": "3.93%"
      },
      "regularMarketVolume": {
        "raw": 6682371,
        "fmt": "6.682M",
        "longFmt": "6,682,371"
      },
      "averageDailyVolume3Month": {
        "raw": 14987160,
```


(continues on next page)

(continued from previous page)


```
        "fmt": "14.987M",
        "longFmt": "14,987,160"
      },
      "marketCap": {
        "raw": 810236992,
        "fmt": "810.237M",
        "longFmt": "810,236,992"
      },
      "fiftyTwoWeekLow": {
        "raw": 1.6,
        "fmt": "1.60"
      },
      "fiftyTwoWeekHigh": {
        "raw": 24.9,
        "fmt": "24.90"
      },
      "regularMarketOpen": {
        "raw": 6.66,
        "fmt": "6.66"
      },
      "priceHint": 2
    },
    {
      "symbol": "HRTX",
      "shortName": "Heron Therapeutics, Inc.",
      "longName": "Heron Therapeutics, Inc.",
      "quoteType": "EQUITY",
      "currency": "USD",
      "regularMarketPrice": {
        "raw": 15.65,
        "fmt": "15.65"
      },
      "regularMarketChange": {
        "raw": 0.029999733,
        "fmt": "0.03"
      },
      "regularMarketChangePercent": {
        "raw": 0.19205976,
        "fmt": "0.19%"
      },
      "regularMarketVolume": {
        "raw": 2699888,
        "fmt": "2.7M",
        "longFmt": "2,699,888"
      },
      "averageDailyVolume3Month": {
        "raw": 1026709,
        "fmt": "1.027M",
        "longFmt": "1,026,709"
      },
      "marketCap": {
        "raw": 1430355072,
        "fmt": "1.43B",
        "longFmt": "1,430,355,072"
      },
      "fiftyTwoWeekLow": {
        "raw": 10.74,
```


(continues on next page)

(continued from previous page)


```
        "fmt": "10.74"
      },
      "fiftyTwoWeekHigh": {
        "raw": 22.4,
        "fmt": "22.40"
      },
      "regularMarketOpen": {
        "raw": 15.73,
        "fmt": "15.73"
      },
      "priceHint": 2
    },
    {
      "symbol": "ARCH",
      "shortName": "Arch Resources, Inc.",
      "longName": "Arch Resources, Inc.",
      "quoteType": "EQUITY",
      "currency": "USD",
      "regularMarketPrice": {
        "raw": 46.54,
        "fmt": "46.54"
      },
      "regularMarketChange": {
        "raw": -0.9300003,
        "fmt": "-0.93"
      },
      "regularMarketChangePercent": {
        "raw": -1.9591326,
        "fmt": "-1.96%"
      },
      "regularMarketVolume": {
        "raw": 674780,
        "fmt": "674,780",
        "longFmt": "674,780"
      },
      "averageDailyVolume3Month": {
        "raw": 386693,
        "fmt": "386,693",
        "longFmt": "386,693"
      },
      "marketCap": {
        "raw": 712848512,
        "fmt": "712.849M",
        "longFmt": "712,848,512"
      },
      "fiftyTwoWeekLow": {
        "raw": 21.8,
        "fmt": "21.80"
      },
      "fiftyTwoWeekHigh": {
        "raw": 58.88,
        "fmt": "58.88"
      },
      "regularMarketOpen": {
        "raw": 47.51,
        "fmt": "47.51"
      },
```


(continues on next page)

(continued from previous page)


```
      "priceHint": 2
    },
    {
      "symbol": "PETS",
      "shortName": "PetMed Express, Inc.",
      "longName": "PetMed Express, Inc.",
      "quoteType": "EQUITY",
      "currency": "USD",
      "regularMarketPrice": {
        "raw": 34,
        "fmt": "34.00"
      },
      "regularMarketChange": {
        "raw": 0.11999893,
        "fmt": "0.12"
      },
      "regularMarketChangePercent": {
        "raw": 0.35418808,
        "fmt": "0.35%"
      },
      "regularMarketVolume": {
        "raw": 780790,
        "fmt": "780,790",
        "longFmt": "780,790"
      },
      "averageDailyVolume3Month": {
        "raw": 868904,
        "fmt": "868,904",
        "longFmt": "868,904"
      },
      "marketCap": {
        "raw": 689217408,
        "fmt": "689.217M",
        "longFmt": "689,217,408"
      },
      "trailingPE": {
        "raw": 22.222223,
        "fmt": "22.22"
      },
      "fiftyTwoWeekLow": {
        "raw": 24.8,
        "fmt": "24.80"
      },
      "fiftyTwoWeekHigh": {
        "raw": 57,
        "fmt": "57.00"
      },
      "regularMarketOpen": {
        "raw": 34.08,
        "fmt": "34.08"
      },
      "priceHint": 2
    },
    {
      "symbol": "UNFI",
      "shortName": "United Natural Foods, Inc.",
      "longName": "United Natural Foods, Inc.",
```


(continues on next page)

(continued from previous page)


```
      "quoteType": "EQUITY",
      "currency": "USD",
      "regularMarketPrice": {
        "raw": 34.73,
        "fmt": "34.73"
      },
      "regularMarketChange": {
        "raw": 1.6199989,
        "fmt": "1.62"
      },
      "regularMarketChangePercent": {
        "raw": 4.8927784,
        "fmt": "4.89%"
      },
      "regularMarketVolume": {
        "raw": 1967154,
        "fmt": "1.967M",
        "longFmt": "1,967,154"
      },
      "averageDailyVolume3Month": {
        "raw": 1583036,
        "fmt": "1.583M",
        "longFmt": "1,583,036"
      },
      "marketCap": {
        "raw": 1955160064,
        "fmt": "1.955B",
        "longFmt": "1,955,160,064"
      },
      "trailingPE": {
        "raw": 9.922857,
        "fmt": "9.92"
      },
      "fiftyTwoWeekLow": {
        "raw": 8.28,
        "fmt": "8.28"
      },
      "fiftyTwoWeekHigh": {
        "raw": 41.37,
        "fmt": "41.37"
      },
      "regularMarketOpen": {
        "raw": 33.19,
        "fmt": "33.19"
      },
      "priceHint": 2
    },
    {
      "symbol": "BBBY",
      "shortName": "Bed Bath & Beyond Inc.",
      "longName": "Bed Bath & Beyond Inc.",
      "quoteType": "EQUITY",
      "currency": "USD",
      "regularMarketPrice": {
        "raw": 31.58,
        "fmt": "31.58"
      },
```


(continues on next page)

(continued from previous page)


```
      "regularMarketChange": {
        "raw": 0.9899998,
        "fmt": "0.99"
      },
      "regularMarketChangePercent": {
        "raw": 3.2363513,
        "fmt": "3.24%"
      },
      "regularMarketVolume": {
        "raw": 5879847,
        "fmt": "5.88M",
        "longFmt": "5,879,847"
      },
      "averageDailyVolume3Month": {
        "raw": 13366711,
        "fmt": "13.367M",
        "longFmt": "13,366,711"
      },
      "marketCap": {
        "raw": 3827969792,
        "fmt": "3.828B",
        "longFmt": "3,827,969,792"
      },
      "fiftyTwoWeekLow": {
        "raw": 3.43,
        "fmt": "3.43"
      },
      "fiftyTwoWeekHigh": {
        "raw": 53.9,
        "fmt": "53.90"
      },
      "regularMarketOpen": {
        "raw": 30.52,
        "fmt": "30.52"
      },
      "priceHint": 2
    },
    {
      "symbol": "IRBT",
      "shortName": "iRobot Corporation",
      "longName": "iRobot Corporation",
      "quoteType": "EQUITY",
      "currency": "USD",
      "regularMarketPrice": {
        "raw": 119.15,
        "fmt": "119.15"
      },
      "regularMarketChange": {
        "raw": 0.6800003,
        "fmt": "0.68"
      },
      "regularMarketChangePercent": {
        "raw": 0.5739852,
        "fmt": "0.57%"
      },
      "regularMarketVolume": {
        "raw": 966674,
```


(continues on next page)

(continued from previous page)


```
        "fmt": "966,674",
        "longFmt": "966,674"
      },
      "averageDailyVolume3Month": {
        "raw": 1122757,
        "fmt": "1.123M",
        "longFmt": "1,122,757"
      },
      "marketCap": {
        "raw": 3359922688,
        "fmt": "3.36B",
        "longFmt": "3,359,922,688"
      },
      "trailingPE": {
        "raw": 23.180935,
        "fmt": "23.18"
      },
      "fiftyTwoWeekLow": {
        "raw": 37.01,
        "fmt": "37.01"
      },
      "fiftyTwoWeekHigh": {
        "raw": 197.4,
        "fmt": "197.40"
      },
      "regularMarketOpen": {
        "raw": 118.61,
        "fmt": "118.61"
      },
      "priceHint": 2
    },
    {
      "symbol": "GPRE",
      "shortName": "Green Plains, Inc.",
      "longName": "Green Plains Inc.",
      "quoteType": "EQUITY",
      "currency": "USD",
      "regularMarketPrice": {
        "raw": 25.63,
        "fmt": "25.63"
      },
      "regularMarketChange": {
        "raw": 0.54999924,
        "fmt": "0.55"
      },
      "regularMarketChangePercent": {
        "raw": 2.1929796,
        "fmt": "2.19%"
      },
      "regularMarketVolume": {
        "raw": 1207272,
        "fmt": "1.207M",
        "longFmt": "1,207,272"
      },
      "averageDailyVolume3Month": {
        "raw": 1073568,
        "fmt": "1.074M",
```


(continues on next page)

(continued from previous page)


```
        "longFmt": "1,073,568"
      },
      "marketCap": {
        "raw": 1108781952,
        "fmt": "1.109B",
        "longFmt": "1,108,781,952"
      },
      "fiftyTwoWeekLow": {
        "raw": 3.81,
        "fmt": "3.81"
      },
      "fiftyTwoWeekHigh": {
        "raw": 28.77,
        "fmt": "28.77"
      },
      "regularMarketOpen": {
        "raw": 23.81,
        "fmt": "23.81"
      },
      "priceHint": 2
    },
    {
      "symbol": "GME",
      "shortName": "GameStop Corporation",
      "longName": "GameStop Corp.",
      "quoteType": "EQUITY",
      "currency": "USD",
      "regularMarketPrice": {
        "raw": 202.44,
        "fmt": "202.44"
      },
      "regularMarketChange": {
        "raw": 0.69000244,
        "fmt": "0.69"
      },
      "regularMarketChangePercent": {
        "raw": 0.34200862,
        "fmt": "0.34%"
      },
      "regularMarketVolume": {
        "raw": 24677297,
        "fmt": "24.677M",
        "longFmt": "24,677,297"
      },
      "averageDailyVolume3Month": {
        "raw": 44219059,
        "fmt": "44.219M",
        "longFmt": "44,219,059"
      },
      "marketCap": {
        "raw": 14119582720,
        "fmt": "14.12B",
        "longFmt": "14,119,582,720"
      },
      "fiftyTwoWeekLow": {
        "raw": 2.57,
        "fmt": "2.57"
```


(continues on next page)

(continued from previous page)


```
    },
    "fiftyTwoWeekHigh": {
      "raw": 483,
      "fmt": "483.00"
    },
    "regularMarketOpen": {
      "raw": 195.73,
      "fmt": "195.73"
    },
    "priceHint": 2
  },
  {
    "symbol": "SRNE",
    "shortName": "Sorrento Therapeutics, Inc.",
    "longName": "Sorrento Therapeutics, Inc.",
    "quoteType": "EQUITY",
    "currency": "USD",
    "regularMarketPrice": {
      "raw": 9.56,
      "fmt": "9.56"
    },
    "regularMarketChange": {
      "raw": 0.11000061,
      "fmt": "0.11"
    },
    "regularMarketChangePercent": {
      "raw": 1.1640277,
      "fmt": "1.16%"
    },
    "regularMarketVolume": {
      "raw": 14339713,
      "fmt": "14.34M",
      "longFmt": "14,339,713"
    },
    "averageDailyVolume3Month": {
      "raw": 18243480,
      "fmt": "18.243M",
      "longFmt": "18,243,480"
    },
    "marketCap": {
      "raw": 2685031168,
      "fmt": "2.685B",
      "longFmt": "2,685,031,168"
    },
    "fiftyTwoWeekLow": {
      "raw": 1.7,
      "fmt": "1.70"
    },
    "fiftyTwoWeekHigh": {
      "raw": 19.39,
      "fmt": "19.39"
    },
    "regularMarketOpen": {
      "raw": 9.4584,
      "fmt": "9.46"
    },
    "priceHint": 2
```


(continues on next page)

(continued from previous page)


```
    },
    {
      "symbol": "OMER",
      "shortName": "Omeros Corporation",
      "longName": "Omeros Corporation",
      "quoteType": "EQUITY",
      "currency": "USD",
      "regularMarketPrice": {
        "raw": 19.91,
        "fmt": "19.91"
      },
      "regularMarketChange": {
        "raw": 0.9699993,
        "fmt": "0.97"
      },
      "regularMarketChangePercent": {
        "raw": 5.1214323,
        "fmt": "5.12%"
      },
      "regularMarketVolume": {
        "raw": 695530,
        "fmt": "695,530",
        "longFmt": "695,530"
      },
      "averageDailyVolume3Month": {
        "raw": 775257,
        "fmt": "775,257",
        "longFmt": "775,257"
      },
      "marketCap": {
        "raw": 1233101952,
        "fmt": "1.233B",
        "longFmt": "1,233,101,952"
      },
      "fiftyTwoWeekLow": {
        "raw": 9.25,
        "fmt": "9.25"
      },
      "fiftyTwoWeekHigh": {
        "raw": 25.46,
        "fmt": "25.46"
      },
      "regularMarketOpen": {
        "raw": 18.93,
        "fmt": "18.93"
      },
      "priceHint": 2
    },
    {
      "symbol": "TMBR",
      "shortName": "Timber Pharmaceuticals, Inc.",
      "longName": "Timber Pharmaceuticals, Inc.",
      "quoteType": "EQUITY",
      "currency": "USD",
      "regularMarketPrice": {
        "raw": 2.24,
        "fmt": "2.2400"
```


(continues on next page)

(continued from previous page)


```
      },
      "regularMarketChange": {
        "raw": 0.01999998,
        "fmt": "0.0200"
      },
      "regularMarketChangePercent": {
        "raw": 0.9009,
        "fmt": "0.90%"
      },
      "regularMarketVolume": {
        "raw": 3054710,
        "fmt": "3.055M",
        "longFmt": "3,054,710"
      },
      "averageDailyVolume3Month": {
        "raw": 11088947,
        "fmt": "11.089M",
        "longFmt": "11,088,947"
      },
      "marketCap": {
        "raw": 26541760,
        "fmt": "26.542M",
        "longFmt": "26,541,760"
      },
      "fiftyTwoWeekLow": {
        "raw": 0.665,
        "fmt": "0.6650"
      },
      "fiftyTwoWeekHigh": {
        "raw": 10.44,
        "fmt": "10.4400"
      },
      "regularMarketOpen": {
        "raw": 2.23,
        "fmt": "2.2300"
      },
      "priceHint": 4
    },
    {
      "symbol": "TRHC",
      "shortName": "Tabula Rasa HealthCare, Inc.",
      "longName": "Tabula Rasa HealthCare, Inc.",
      "quoteType": "EQUITY",
      "currency": "USD",
      "regularMarketPrice": {
        "raw": 44.51,
        "fmt": "44.51"
      },
      "regularMarketChange": {
        "raw": 0.11999893,
        "fmt": "0.12"
      },
      "regularMarketChangePercent": {
        "raw": 0.27032876,
        "fmt": "0.27%"
      },
      "regularMarketVolume": {
```


(continues on next page)

(continued from previous page)


```
        "raw": 678554,
        "fmt": "678,554",
        "longFmt": "678,554"
      },
      "averageDailyVolume3Month": {
        "raw": 474606,
        "fmt": "474,606",
        "longFmt": "474,606"
      },
      "marketCap": {
        "raw": 1098617984,
        "fmt": "1.099B",
        "longFmt": "1,098,617,984"
      },
      "fiftyTwoWeekLow": {
        "raw": 30.12,
        "fmt": "30.12"
      },
      "fiftyTwoWeekHigh": {
        "raw": 69.31,
        "fmt": "69.31"
      },
      "regularMarketOpen": {
        "raw": 44.33,
        "fmt": "44.33"
      },
      "priceHint": 2
    },
    {
      "symbol": "BGS",
      "shortName": "B&G Foods, Inc. B&G Foods, Inc.",
      "longName": "B&G Foods, Inc.",
      "quoteType": "EQUITY",
      "currency": "USD",
      "regularMarketPrice": {
        "raw": 33.06,
        "fmt": "33.06"
      },
      "regularMarketChange": {
        "raw": 1.3400021,
        "fmt": "1.34"
      },
      "regularMarketChangePercent": {
        "raw": 4.2244706,
        "fmt": "4.22%"
      },
      "regularMarketVolume": {
        "raw": 2256120,
        "fmt": "2.256M",
        "longFmt": "2,256,120"
      },
      "averageDailyVolume3Month": {
        "raw": 2242804,
        "fmt": "2.243M",
        "longFmt": "2,242,804"
      },
      "marketCap": {
```


(continues on next page)

(continued from previous page)


```
        "raw": 2139778816,
        "fmt": "2.14B",
        "longFmt": "2,139,778,816"
      },
      "trailingPE": {
        "raw": 16.205883,
        "fmt": "16.21"
      },
      "fiftyTwoWeekLow": {
        "raw": 15.65,
        "fmt": "15.65"
      },
      "fiftyTwoWeekHigh": {
        "raw": 47.84,
        "fmt": "47.84"
      },
      "regularMarketOpen": {
        "raw": 31.89,
        "fmt": "31.89"
      },
      "priceHint": 2
    }
  ],
  "columns": [
    {
      "data": "symbol"
    },
    {
      "data": "shortName"
    },
    {
      "data": "longName"
    },
    {
      "data": "quoteType"
    },
    {
      "data": "currency"
    },
    {
      "data": "currencyCode"
    },
    {
      "data": "underlyingSymbol"
    },
    {
      "data": "regularMarketPrice"
    },
    {
      "data": "regularMarketChange"
    },
    {
      "data": "regularMarketChangePercent"
    },
    {
      "data": "regularMarketVolume"
    },
```


(continues on next page)

(continued from previous page)


```
    {
      "data": "averageDailyVolume3Month"
    },
    {
      "data": "marketCap"
    },
    {
      "data": "trailingPE"
    },
    {
      "data": "fiftyTwoWeekLow"
    },
    {
      "data": "fiftyTwoWeekHigh"
    },
    {
      "data": "regularMarketOpen"
    }
  ],
  "total": 4422,
  "filters": {}
}
```


class virtual_finance_api.endpoints.yahoo.**Screeners**

Bases: virtual_finance_api.endpoints.apirequest.VirtualAPIRequest

Screeners - class to handle the screeners endpoint.

DOMAIN = 'https://finance.yahoo.com'

ENDPOINT = 'screener'

EXPECTED_STATUS = 200

METHOD = 'GET'

RESPONSE_TYPE = 'txt'

__init__() → None

Instantiate a Screeners APIRequest instance.

returns the available prefined screeners.

```
>>> import virtual_finance_api as fa
>>> import virtual_finance_api.endpoints.yahoo as yh
>>> client = fa.Client()
>>> r = yh.Screeners()
>>> rv = client.request(r)
>>> print(json.dumps(r.response, indent=2))
```


```
{
  "screeners": [
    {
      "title": "Fair Value Screener",
      "predefinedScr": true,
      "description": "Undervalued stocks with a strong & consistent history␣
↪of earnings and revenue growth, sorted by rate of return",
      "canonicalName": "FAIR_VALUE_SCREENER"
    },
```


(continues on next page)

(continued from previous page)


```
    {
      "title": "Most Shorted Stocks",
      "predefinedScr": true,
      "description": "Stocks with the highest short interest positions from␣
↪Nasdaq and NYSE reports released every two weeks.",
      "canonicalName": "MOST_SHORTED_STOCKS"
    },
    {
      "title": "Undervalued Growth Stocks",
      "predefinedScr": true,
      "description": "Stocks with earnings growth rates better than 25% and␣
↪relatively low PE and PEG ratios.",
      "canonicalName": "UNDERVALUED_GROWTH_STOCKS"
    },
    {
      "title": "Growth Technology Stocks",
      "predefinedScr": true,
      "description": "Technology stocks with revenue and earnings growth in␣
↪excess of 25%.",
      "canonicalName": "GROWTH_TECHNOLOGY_STOCKS"
    },
    {
      "title": "Day Gainers - US",
      "predefinedScr": true,
      "description": "Stocks ordered in descending order by price percent␣
↪change greater than 3% with respect to the previous close",
      "canonicalName": "DAY_GAINERS"
    },
    {
      "title": "Day Losers - US",
      "predefinedScr": true,
      "description": "Stocks ordered in ascending order by price percent␣
↪change with respect to the previous close",
      "canonicalName": "DAY_LOSERS"
    },
    {
      "title": "Most Actives - US",
      "predefinedScr": true,
      "description": "Stocks ordered in descending order by intraday trade␣
↪volume",
      "canonicalName": "MOST_ACTIVES"
    },
    {
      "title": "Potentially undervalued large cap stocks",
      "predefinedScr": true,
      "description": "Large cap stocks that are potentially undervalued,␣
↪ordered descending by volume",
      "canonicalName": "UNDERVALUED_LARGE_CAPS"
    },
    {
      "title": "Aggresive small cap stocks",
      "predefinedScr": true,
      "description": "Small cap stocks with high earnings growth rates",
      "canonicalName": "AGGRESSIVE_SMALL_CAPS"
    },
    {
      "title": "Small caps with momentum",
```


(continues on next page)

(continued from previous page)


```
      "predefinedScr": true,
      "description": "Small cap stocks with percentchange greater than 5%",
      "canonicalName": "SMALL_CAP_GAINERS"
    },
    {
      "title": "Top Mutual Funds",
      "predefinedScr": true,
      "description": "Funds with Performance Rating of 4 & 5 ordered by␣
↪Percent Change",
      "canonicalName": "TOP_MUTUAL_FUNDS"
    },
    {
      "title": "Portfolio Anchors",
      "predefinedScr": true,
      "description": "Funds with Performance Rating of 4 & 5 and top-half␣
↪returns that could serve as a rock-solid core of an investor&#39;s portfolio
↪",
      "canonicalName": "PORTFOLIO_ANCHORS"
    },
    {
      "title": "Solid Large Growth Funds",
      "predefinedScr": true,
      "description": "Large Growth Funds with Performance Rating of 4 & 5 and␣
↪top-half returns",
      "canonicalName": "SOLID_LARGE_GROWTH_FUNDS"
    },
    {
      "title": "Solid Mid-Cap Growth Funds",
      "predefinedScr": true,
      "description": "Mid-Cap Growth Funds with Performance Rating of 4 & 5␣
↪and top-half returns",
      "canonicalName": "SOLID_MIDCAP_GROWTH_FUNDS"
    },
    {
      "title": "Conservative Foreign Funds",
      "predefinedScr": true,
      "description": "Foreign funds with Performance Rating of 4 & 5, low␣
↪risk and top-half returns",
      "canonicalName": "CONSERVATIVE_FOREIGN_FUNDS"
    },
    {
      "title": "High Yield Bond",
      "predefinedScr": true,
      "description": "High Yield Bond with Performance Rating of 4 & 5, low␣
↪risk and top-half returns",
      "canonicalName": "HIGH_YIELD_BOND"
    }
  ]
}
```


class virtual_finance_api.endpoints.yahoo.**YhooIndex**(*index: str*)

Bases: virtual_finance_api.endpoints.apirequest.VirtualAPIRequest

YhooIndex - request class to handle the index overview endpoint.

DOMAIN = 'https://finance.yahoo.com'

ENDPOINT = 'quote/{index}/components'

EXPECTED_STATUS = 200

METHOD = 'GET'

RESPONSE_TYPE = 'txt'

__init__(*index: str*) → None
Instantiate a YhooIndex APIRequest instance.

Parameters **index** (*string (required)*) – the ticker of the index to perform the request for.

Example

```
>>> import virtual_finance_api as fa
>>> import virtual_finance_api.endpoints.yahoo as yh
>>> client = fa.Client()
>>> r = yh.YhooIndex('^IXIC')
>>> rv = client.request(r)
>>> print(json.dumps(rv, indent=2))
```


```
{
  "Symbol": {
    "0": "TWST",
    "1": "SLAB",
    "2": "SCOA",
    "3": "STRT",
    "4": "NEBC",
    "5": "FEYE",
    "6": "NVCN",
    "7": "NVCR",
    "8": "IFRX",
    "9": "HLAH",
    "10": "BBI",
    "11": "OPTN",
    "12": "FNLC",
    "13": "SDC",
    "14": "ESXB",
    "15": "KLXE",
    "16": "OPTT",
    "17": "CVCY",
    "18": "CVCO",
    "19": "BBQ",
    "20": "NMRD",
    "21": "NVEC",
    "22": "SCOR",
    "23": "STRS",
    "24": "SCR",
    "25": "NMRK",
    "26": "CMRX",
    "27": "SDH",
    "28": "FNKO",
    "29": "AGLE"
  },
  "Company Name": {
    "0": "Twist Bioscience Corporation",
    "1": "Silicon Laboratories Inc.",
```


(continues on next page)

(continued from previous page)


```
    "2": "Scion Tech Growth I",
    "3": "Strattec Security Corporation",
    "4": "Nebula Caravel Acquisition Corp.",
    "5": "FireEye, Inc.",
    "6": "Neovasc Inc.",
    "7": "NovoCure Limited",
    "8": "InflaRx N.V.",
    "9": "Hamilton Lane Alliance Holdings I, Inc.",
    "10": "Brickell Biotech, Inc.",
    "11": "OptiNose, Inc.",
    "12": "The First Bancorp, Inc.",
    "13": "SmileDirectClub, Inc.",
    "14": "Community Bankers Trust Corporation",
    "15": "KLX Energy Services Holdings, Inc.",
    "16": "Ocean Power Technologies, Inc.",
    "17": "Central Valley Community Bancorp",
    "18": "Cavco Industries, Inc.",
    "19": "BBQ Holdings, Inc.",
    "20": "Nemaura Medical Inc.",
    "21": "NVE Corporation",
    "22": "comScore, Inc.",
    "23": "Stratus Properties Inc.",
    "24": "Score Media and Gaming Inc.",
    "25": "Newmark Group, Inc.",
    "26": "Chimerix, Inc.",
    "27": "Global Internet of People, Inc.",
    "28": "Funko, Inc.",
    "29": "Aeglea BioTherapeutics, Inc."
  },
  "Last Price": {
    "0": 116.73,
    "1": 136.98,
    "2": 9.71,
    "3": 43.34,
    "4": 9.86,
    "5": 19.76,
    "6": 1.13,
    "7": 127.34,
    "8": 3.89,
    "9": 9.62,
    "10": 1.19,
    "11": 3.53,
    "12": 28.7,
    "13": 10.6,
    "14": 8.52,
    "15": 16.79,
    "16": 2.97,
    "17": 18.93,
    "18": 225.89,
    "19": 7.9,
    "20": 8.02,
    "21": 69.0,
    "22": 3.67,
    "23": 29.95,
    "24": 24.96,
    "25": 10.56,
    "26": 8.94,
```


(continues on next page)

(continued from previous page)


```
    "27": 3.91,
    "28": 20.83,
    "29": 8.13
  },
  "Change": {
    "0": -0.14,
    "1": 0.49,
    "2": -0.04,
    "3": 0.2,
    "4": -0.05,
    "5": 0.1,
    "6": 0.01,
    "7": -1.54,
    "8": -0.06,
    "9": -0.16,
    "10": 0.02,
    "11": 0.06,
    "12": 0.62,
    "13": 0.23,
    "14": 0.25,
    "15": 0.56,
    "16": 0.11,
    "17": 0.72,
    "18": 8.72,
    "19": 0.31,
    "20": -0.36,
    "21": 2.95,
    "22": 0.17,
    "23": -1.58,
    "24": -1.36,
    "25": 0.59,
    "26": 0.5,
    "27": -0.27,
    "28": 1.59,
    "29": 0.79
  },
  "% Change": {
    "0": "-0.12%",
    "1": "+0.36%",
    "2": "-0.41%",
    "3": "+0.46%",
    "4": "-0.50%",
    "5": "+0.51%",
    "6": "+0.89%",
    "7": "-1.19%",
    "8": "-1.52%",
    "9": "-1.64%",
    "10": "+1.71%",
    "11": "+1.73%",
    "12": "+2.21%",
    "13": "+2.22%",
    "14": "+3.02%",
    "15": "+3.45%",
    "16": "+3.85%",
    "17": "+3.95%",
    "18": "+4.02%",
    "19": "+4.08%",
```


(continues on next page)

(continued from previous page)


```
    "20": "-4.30%",
    "21": "+4.47%",
    "22": "+4.86%",
    "23": "-5.01%",
    "24": "-5.17%",
    "25": "+5.92%",
    "26": "+5.92%",
    "27": "-6.46%",
    "28": "+8.26%",
    "29": "+10.76%"
  },
  "Volume": {
    "0": 1445166,
    "1": 268814,
    "2": 51867,
    "3": 24663,
    "4": 706581,
    "5": 2884683,
    "6": 2224508,
    "7": 951071,
    "8": 709346,
    "9": 29364,
    "10": 1338888,
    "11": 324179,
    "12": 11384,
    "13": 3558728,
    "14": 47475,
    "15": 80304,
    "16": 3178991,
    "17": 36728,
    "18": 36770,
    "19": 63289,
    "20": 108626,
    "21": 40170,
    "22": 518671,
    "23": 23291,
    "24": 593852,
    "25": 898139,
    "26": 742752,
    "27": 330357,
    "28": 5673662,
    "29": 179662
  }
}
```


index

class virtual_finance_api.endpoints.yahoo.**AdjustType**

Bases: str, enum.Enum

An enumeration.

auto = 'auto'

back = 'back'

class virtual_finance_api.endpoints.yahoo.**Period**

Bases: str, enum.Enum

An enumeration.

p_10y = '10y'

p_1d = '1d'

p_1mo = '1mo'

p_1y = '1y'

p_2y = '2y'

p_3mo = '3mo'

p_5d = '5d'

p_5y = '5y'

p_6mo = '6mo'

p_max = 'max'

p_ytd = 'ytd'

class virtual_finance_api.endpoints.yahoo.**Interval**

Bases: str, enum.Enum

An enumeration.

i_15m = '15m'

i_1d = '1d'

i_1h = '1h'

i_1m = '1m'

i_1mo = '1mo'

i_1wk = '1wk'

i_2m = '2m'

i_30m = '30m'

i_3mo = '3mo'

i_5d = '5d'

i_5m = '5m'

i_60m = '60m'

i_90m = '90m'

Business Insider endpoints

class virtual_finance_api.endpoints.business_insider.**ISIN**(*params: dict*)

Bases: virtual_finance_api.endpoints.apirequest.VirtualAPIRequest

ISIN - class to handle the ISIN endpoint.

DOMAIN = 'https://markets.businessinsider.com'

ENDPOINT = 'ajax/SearchController_Suggest'

EXPECTED_STATUS = 200

METHOD = 'GET'

RESPONSE_TYPE = 'txt'

__init__(*params: dict*) → None
Instantiate a ISIN equest instance.

Parameters **params** (*dict (required)*) – max_results: 25 (default), query: <ticker>

Raises **ValueError** – in case of a missing *query* parameter.

Example:

params:

```
{
  "query": "IBM"
}
```

```
>>> print(client.request(ISIN(params=params)))
```

```
{
  "ticker": "IBM",
  "ISIN": "US4592001014"
}
```

Compatibility requests: yfinance

This module provides the *yfinance* compatible requests.

These requests are derived from the Yahoo base classes, but all classes provide *yfinance.Ticker* compatible properties. So, some return a dict, some return Pandas series and some return a Pandas dataframe, just like *yfinance* does.

class virtual_finance_api.compat.yfinance.endpoints.**Financials**(*ticker*)
Bases: virtual_finance_api.endpoints.yahoo.ticker_bundle.Financials

Financials - class to handle the financials endpoint.

__init__(*ticker*)
Instantiate a Financials APIRequest instance.

Parameters **ticker** (*string (required)*) – the ticker to perform the request for.

```
>>> import virtual_finance_api as fa
>>> # import the yfinance compatible endpoints
>>> import virtual_finance_api.compat.yfinance.endpoints as yf
>>> client = fa.Client()
>>> r = yf.Financials('IBM')
>>> rv = client.request(r)
```

```
>>> # now we can use the request properties to fetch data
>>> print(r.earnings)
>>> # ... earnings as a dict with Pandas Dataframes equal to yfinance
```

Note: The full response of the parent request is still available in the return value and the response property

```
>>> # the dataframes combined as JSON
>>> yq = dict([(k, json.loads(r.earnings[k].to_json())) for k in ('yearly',
↪'quarterly')])
>>> print(json.dumps(yq, indent=2))
```


```
{
  "yearly": {
    "Revenue": {
      "2017": 79139000000,
      "2018": 79591000000,
      "2019": 77147000000,
      "2020": 73621000000
    },
    "Earnings": {
      "2017": 5753000000,
      "2018": 8728000000,
      "2019": 9431000000,
      "2020": 5590000000
    }
  },
  "quarterly": {
    "Revenue": {
      "1Q2020": 17571000000,
      "2Q2020": 18121000000,
      "3Q2020": 17561000000,
      "4Q2020": 20368000000
    },
    "Earnings": {
      "1Q2020": 1175000000,
      "2Q2020": 1361000000,
      "3Q2020": 1698000000,
      "4Q2020": 1356000000
    }
  }
}
```


balancesheet

cashflow

earnings

financials

class virtual_finance_api.compat.yfinance.endpoints.**History**(*ticker*, *params*)

Bases: virtual_finance_api.endpoints.yahoo.ticker_bundle.History

History - class to handle the history endpoint.

__init__(*ticker*, *params*)

Instantiate a History APIRequest instance.

Parameters

- **ticker** (*string (required)*) – the ticker to perform the request for.
- **params** (*dict (optional)*) – dictionary with optional parameters to perform the request parameters default to 1 month of daily (1d) historical data.

```
{
  "range": "max",
  "interval": "1d",
  "events": "div,splits"
}
```

```
>>> import virtual_finance_api as fa
>>> import virtual_finance_api.compat.yfinance.endpoints as yf
>>> client = fa.Client()
>>> r = yf.History('IBM', params=params)
>>> rv = client.request(r)
```

```
>>> # now we can use the request properties to fetch data
>>> print(r.history)
>>> # ... the history as a Pandas Dataframe
```

```
{}
```

actions

dividends

history

splits

class virtual_finance_api.compat.yfinance.endpoints.**Holders**(*ticker*)

Bases: virtual_finance_api.endpoints.yahoo.ticker_bundle.Holders

Holders - class to handle the holders endpoint.

__init__(*ticker*)

Instantiate a Holders APIRequest instance.

Parameters **ticker** (*string (required)*) – the ticker to perform the request for.

```
>>> import virtual_finance_api as fa
>>> import virtual_finance_api.compat.yfinance.endpoints as yf
>>> client = fa.Client()
>>> r = yf.Holders('IBM')
>>> rv = client.request(r)
```

```
>>> # now we can use the request properties to fetch data
>>> print(r.majors)
>>> # ... the majors as a Pandas Dataframe equal to yfinance
```

```
>>> # the JSON representation of the dataframe
>>> print(r.majors.to_json())
```

```
{
  "0": {
    "0": "0.13%",
    "1": "58.58%",
    "2": "58.66%",
    "3": "2561"
  },
  "1": {
```

(continues on next page)

(continued from previous page)

```
    "0": "% of Shares Held by All Insider",
    "1": "% of Shares Held by Institutions",
    "2": "% of Float Held by Institutions",
    "3": "Number of Institutions Holding Shares"
  }
}
```

institutional

major

mutualfund

class virtual_finance_api.compat.yfinance.endpoints.**Options**(*ticker*, *params=None*)

Bases: virtual_finance_api.endpoints.yahoo.ticker_bundle.Options

Options - class to handle the options endpoint.

__init__(*ticker*, *params=None*)

Instantiate a Profile APIRequest instance.

Parameters **ticker** (*string (required)*) – the ticker to perform the request for.

```
>>> import virtual_finance_api as fa
>>> # import the yfinance compatible endpoints
>>> import virtual_finance_api.compat.yfinance.endpoints as yf
>>> client = fa.Client()
>>> r = yf.Options('IBM')
>>> rv = client.request(r)
```

```
>>> # now we can use the request properties to fetch data
>>> print(r.options)
>>> # ... all the expiration dates
```

```
[
  "2021-03-26",
  "2021-04-01",
  "2021-04-09",
  "2021-04-16",
  "2021-04-23",
  "2021-04-30",
  "2021-05-21",
  "2021-06-18",
  "2021-07-16",
  "2021-09-17",
  "2021-10-15",
  "2022-01-21",
  "2023-01-20"
]
```

```
>>> # and, just like yfinance: the dataframes with calls and puts
>>> print(r.option_chain('2021-03-26')[0]  # all calls
>>> print(r.option_chain('2021-03-26')[1]  # all puts
```

option_chain(*date=None*, *proxy=None*, *tz=None*)

option_chain - return option chain dataframes for calls/puts.

options

class virtual_finance_api.compat.yfinance.endpoints.**Profile**(*ticker*)
Bases: virtual_finance_api.endpoints.yahoo.ticker_bundle.Profile

Profile - class to handle the profile endpoint.

COMPONENTS = ['defaultKeyStatistics', 'details', 'summaryProfile', 'recommendationTren

Calendar()

Info()

Recommendations()

Sustainability()

__init__(*ticker*)
Instantiate a Profile APIRequest instance.

Parameters **ticker** (*string (required)*) – the ticker to perform the request for.

```
>>> import virtual_finance_api as fa
>>> # import the yfinance compatible endpoints
>>> import virtual_finance_api.compat.yfinance.endpoints as yf
>>> client = fa.Client()
>>> r = yf.Profile('IBM')
>>> rv = client.request(r)
```

```
>>> # now we can use the request properties to fetch data
>>> print(r.calendar)
>>> # ... the calendar as a Pandas Dataframe
```

```
>>> # the JSON representation of the dataframe
>>> print(r.calendar.to_json())
```

```
{
  "Value": {
    "Earnings Date": 1618790400000,
    "Earnings Average": 1.63,
    "Earnings Low": 1.39,
    "Earnings High": 1.82,
    "Revenue Average": 17377200000,
    "Revenue Low": 17108000000,
    "Revenue High": 17607300000
  }
}
```

calendar

info

recommendations

sustainability

yfinance Ticker compatibility class

The Ticker-class aims to be the compatible counterpart of yfinance.Ticker. The Ticker class basically wraps all the requests needed to represent all the properties like yfinance.Ticker.

This class is provided for compatibility reasons. If you only need certain data, the advise is to use the request providing that data.

If you still want the yfinance compatible output you can use one of the *compat.yfinance.endpoints* request classes.

The other option is to use one tof the *extensions.stdjson* request classes. These classes provide a standardized JSON response.

class virtual_finance_api.compat.yfinance.**Ticker**(*ticker*)

Bases: object

__init__(*ticker*)

Instantiate a Ticker instance.

Parameters **ticker** (*string (required)*) – the ticker to perform the request for.

The constructor will instantiate all the requests needed to fetch data for certain properties. But the requests will only be executed when the data is asked for.

```
>>> import json
>>> import virtual_finance_api.compat.yfinance as yf
>>> t = yf.Ticker('IBM')
```

```
>>> # now we can use the request properties to fetch data
>>> print(t.earnings)
>>> print(t.quarterly_earnings)
>>> # ... earnings as a dict with Pandas Dataframes equal to yfinance
```

```
>>> # the dataframes combined as JSON
>>> yq = dict([(k, json.loads(getattr(t, k).to_json())) for k in ('earnings',
↪'quarterly_earnings')])
>>> print(json.dumps(yq, indent=2))
```

```
{
  "yearly": {
    "Revenue": {
      "2017": 79139000000,
      "2018": 79591000000,
      "2019": 77147000000,
      "2020": 73621000000
    },
    "Earnings": {
      "2017": 5753000000,
      "2018": 8728000000,
      "2019": 9431000000,
      "2020": 5590000000
    }
  },
  "quarterly": {
    "Revenue": {
      "1Q2020": 17571000000,
      "2Q2020": 18121000000,
      "3Q2020": 17561000000,
      "4Q2020": 20368000000
    },
    "Earnings": {
      "1Q2020": 1175000000,
      "2Q2020": 1361000000,
      "3Q2020": 1698000000,
      "4Q2020": 1356000000
    }
```

(continues on next page)

(continued from previous page)

```
    }
}
```

actions

balance_sheet

balancesheet

calendar

cashflow

dividends

earnings

financials

history(***kwargs*)

info

institutional_holders

isin

major_holders

mutualfund_holders

option_chain(*date*)

options

quarterly_balance_sheet

quarterly_balancesheet

quarterly_cashflow

quarterly_earnings

quarterly_financials

recommendations

splits

sustainability

Indices and tables

- genindex
- modindex
- search

1.4.5 Changelog

[Unreleased]

v0.6.0 (2021-06-01)

New Features

- [types] ReportPeriod added

Bug Fixes

- [History] handle adjustment of data correct

Refactoring

- [cli] use typer instead of click
- option processing regarding History requests split hprocopt into procopt for the yahoo History request and yfprocopt for the yfinance compat. History request

Administration and Chores

- [requirements] update dependencies

v0.5.0 (2021-05-30)

New Features

- [types] types Period, Interval added

Refactoring

- [cli] type hints, apply yahoo types
- [tests] switch to pytest from unittest
- [cli] rename entrypoint to vfapi

Documentation Changes

- [README] minor changes/extension in install

Administration and Chores

- [setup] correct URL to repository

v0.4.3 (2021-05-17)

Documentation Changes

- [README] add codestyle 'black' badge

Administration and Chores

- [setup.py] delete 'version=' from version

v0.4.2 (2021-05-17)

Bug Fixes

- [yahoo.Profile] add a 404

Refactoring

- apply typing

v0.4.1 (2021-04-15)

Style Fixes

- [black] black applied to library + tests + setup.py

Refactoring

- [endpoints] make the basic requests return the standardized JSON of the extensions.stdjson requests. Remove the extensions.stdjson. Modify compat.yfinance to use the standardized JSON.

Administration and Chores

- [pre-commit] added / configured pre-commit

v0.4.0 (2021-04-13)

New Features

- [endpoints] extensions.stdjson, a standardized JSON layer

Tests

- [unittest] unittests regarding virtual_finance_api.extensions.stdjson virtual_finance_api.client
- [unittest] yfinance Ticker class tests extended

Bug Fixes

- [Makefile] prevent link creation by pandoc
- [compat.yfinance] Ticker property code fixed for properties: dividends splits actions

Style Fixes

- minor EOL whitespace / empty line changes

Refactoring

- [enpoints] History, Options now VirtualAPIRequests

Documentation Changes

- [endpoints] extensions.stdjson added

Administration and Chores

- [CHANGELOG] templates to generate CHANGELOG

v0.3.2 (2021-03-30)

Administration and Chores

- [requirements] python 3.6 up to pandas 1.1.5

v0.3.1 (2021-03-30)

Documentation Changes

- [README] badges added
- [README] extended with various components

Administration and Chores

- [config] setup.py include requirements correctly
- [travis] fix deployment to pypi

v0.3.0 (2021-03-30)

New Features

- [endpoints] yfinance compatibility endpoints

Tests

- [unitttest] yfinance compatible endpoint tests

Bug Fixes

- [docs] requirements_dev: missing packages

Refactoring

- [endpoints] use rapidjson instead of json

Administration and Chores

- [config] requirements, Makefile update requirements: include rapidjson Makefile extended
- [config] update travis / tox config

v0.2.2 (2021-03-27)

Bug Fixes

- [docs] fix sphinx build

v0.2.1 (2021-03-27)

Documentation Changes

- [sphinx] initial documentation setup
- [README] example added

v0.2.0 (2021-03-26)

New Features

- [yahoo endpoints] Yahoo endpoint request classes
- [endpoints] business_insider ISIN request class
- [generic] ISINCode class to handle ISIN-codes
- [base] base classes classes to handle and setup API requests

Tests

- [yahoo endpoints] unittests for yahoo endpoints
- [unittests] test business_insider endpoint(s)
- [unittest] tests to test Client and generic module

Administration and Chores

- [config] setup travis for coverage, add badges to README.rst
- [config] setup.py and requirements
- [travis] removed unsupported python 3.5
- [config] fix tox config
- [requirements] packages added

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

v

Index

Symbols

A

B

C

D

E

F

H

I

M

O

P