Ran Aroussi [aroussi.com]
@aroussi

Reliably download historical market data from with Python

17 APRIL 2019 RAN AROUSSI #PYTHON #YFINANCE

Ever since Yahoo decommissioned their historical data API, Python developers looked for a reliable workaround. As a result, my library, **yfinance** [https://github.com/ranaroussi/yfinance], gained momentum and ~~was downloaded over 100,000~~ enjoys 300k+ installs per month, acording to PyPi!

Legal note:

Yahoo!, Y!Finance, and Yahoo! finance are registered trademarks of Yahoo, Inc.

yfinance is not affiliated, endorsed, or vetted by Yahoo, Inc. It's an open-source tool that uses Yahoo's publicly available APIs, and is intended for research and educational purposes.

You should refer to Yahoo!'s terms of use (here [https://policies.yahoo.com/us/en/yahoo/terms/product-atos/apiforydn/index.htm], here [https://legal.yahoo.com/us/en/yahoo/terms/otos/index.html], and here [https://policies.yahoo.com/us/en/yahoo/terms/index.htm]) for details on your rights to use the actual data downloaded. Remember - the Yahoo! finance API is intended for personal use only.

`fix-yahoo-finance` **aimed to offer a temporary fix** [https://aroussi.com/post/fix-yahoo-finance] to the problem by getting data from Yahoo! Finance and returning it in the same format as **pandas_datareader's** `get_data_yahoo()`, thus keeping the code changes in exisiting software to minimum.

The problem was, that this hack was a bit unreliable, causing data to not being downloaded and required developers to force session re-initialization and re-fetching of cookies, by calling `yf.get_yahoo_crumb(force=True)`.

yfinance is a complete re-write of the libray, offering a reliable method of downloading historical market data from Yahoo! Finance's API, up to 1 minute granularity, in a more Pythonic way.

Introducing the `Ticker()` module:

The `Ticker()` module allows you get market and meta data for a security, using a Pythonic way:

```
import yfinance as yf

msft = yf.Ticker("MSFT")
print(msft)
"""
returns
<yfinance.Ticker object at 0x1a1715e898>
"""

# get stock info
```

```
msft.info

"""
returns:
{
 'quoteType': 'EQUITY',
 'quoteSourceName': 'Nasdaq Real Time Price',
 'currency': 'USD',
 'shortName': 'Microsoft Corporation',
 'exchangeTimezoneName': 'America/New_York',
  ...
 'symbol': 'MSFT'
}
"""

# get historical market data
msft.history(period="max")
"""
returns:
              Open    High    Low    Close      Volume  Dividends  Splits
Date
1986-03-13    0.06    0.07    0.06    0.07  1031788800        0.0     0.0
1986-03-14    0.07    0.07    0.07    0.07   308160000        0.0     0.0
...
2019-04-15  120.94  121.58  120.57  121.05    15792600        0.0     0.0
2019-04-16  121.64  121.65  120.10  120.77    14059700        0.0     0.0
"""

# show actions (dividends, splits)
msft.actions
"""
returns:
            Dividends  Splits
Date
1987-09-21       0.00     2.0
1990-04-16       0.00     2.0
...
2018-11-14       0.46     0.0
2019-02-20       0.46     0.0
"""

# show dividends
msft.dividends
"""
returns:
Date
2003-02-19    0.08
2003-10-15    0.16
...
2018-11-14    0.46
2019-02-20    0.46
"""

# show splits
msft.splits
"""
returns:
Date
1987-09-21    2.0
1990-04-16    2.0
...
1999-03-29    2.0
2003-02-18    2.0
"""
```

Available paramaters for the `history()` method are:

- **period**: data period to download (Either Use period parameter or use start and end) Valid periods are: 1d, 5d, 1mo, 3mo, 6mo, 1y, 2y, 5y, 10y, ytd, max
- **interval**: data interval (intraday data cannot extend last 60 days) Valid intervals are: 1m, 2m, 5m, 15m, 30m, 60m, 90m, 1h, 1d, 5d, 1wk, 1mo, 3mo
- **start**: If not using period - Download start date string (YYYY-MM-DD) or datetime.
- **end**: If not using period - Download end date string (YYYY-MM-DD) or datetime.
- **prepost**: Include Pre and Post market data in results? (Default is `False`)
- **auto_adjust**: Adjust all OHLC automatically? (Default is `True`)
- **actions**: Download stock dividends and stock splits events? (Default is `True`)

Mass download of market data:

You can also download data for multiple tickers at once, like before.

```
import yfinance as yf
data = yf.download("SPY AAPL", start="2017-01-01", end="2017-04-30")
```

To access the closing price data for **SPY**, you should use: `data['Close']['SPY']`.

If, however, you want to group data by Symbol, use:

```
import yfinance as yf
data = yf.download("SPY AAPL", start="2017-01-01", end="2017-04-30",
                   group_by="ticker")
```

To access the closing price data for **SPY**, you should use: `data['SPY']['Close']`.

The `download()` method accepts an additional parameter - `threads` for faster completion when downloading a lot of symbols at once.

*** NOTE:** To keep compatibility with older versions, **auto_adjust** defaults to `False` when using mass-download.

Using pandas_datareader:

If your legacy code is using `pandas_datareader` and you wand to keep the code changes to minimum, you can simply call the override method and keep your code as it was:

```
from pandas_datareader import data as pdr

import yfinance as yf
yf.pdr_override() # <== that's all it takes :-)

# download dataframe using pandas_datareader
data = pdr.get_data_yahoo("SPY", start="2017-01-01", end="2017-04-30")
```

To install/upgrade **yfinance** using pip, run:

```
$ pip install yfinance --upgrade --no-cache-dir
```

The **Github repository** `[https://github.com/ranaroussi/yfinance]` has more information and issue tracking.

Enjoy!

Updated on 17 April 2019. For the latest version and comments, please see:
https://aroussi.com/post/python-yahoo-finance