yfinance 0.2.36

Latest version

```
pip install yfinance
```

Released: Jan 21, 2024

Download market data from Yahoo! Finance API

Project description

Download market data from Yahoo! Finance's API

***** IMPORTANT LEGAL DISCLAIMER *****

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, here, and here) **for details on your rights to use the actual data downloaded. Remember - the Yahoo! finance API is intended for personal use**

Statistics

GitHub statistics:

- **Stars:** 11339
- **Forks:** 2137
- **Open issues:** 164
- **Open PRs:** 22

View statistics for this project via Libraries.io, or by using our public dataset on Google BigQuery

Meta

License: Apache Software License (Apache)

Author: Ran Aroussi

pandas, yahoo finance, pandas datareader

Maintainers

ranaroussi

Classifiers

only.

---

python 2.7, 3.6+ | pypi v0.2.36 | status beta | installs 1.5M/month | build no longer available | codefactor A | Star 11k | Follow

yfinance offers a threaded and Pythonic way to download market data from Yahoo!® finance.

→ Check out this Blog post for a detailed tutorial with code examples.

Changelog »

Installation

Install `yfinance` using `pip`:

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

With Conda.

To install with optional dependencies, replace `optional` with: `nospam` for caching-requests, `repair` for price repair, or `nospam,repair` for both:

```
$ pip install "yfinance[optional]"
```

Required dependencies , all dependencies.

Development Status

- 4 - Beta

Intended Audience

- Developers

License

- OSI Approved :: Apache Software License

Operating System

- OS Independent

Programming Language

- Python :: 3.6
- Python :: 3.7
- Python :: 3.8
- Python :: 3.9
- Python :: 3.10

Topic

- Office/Business :: Financial
- Office/Business :: Financial :: Investment
- Scientific/Engineering :: Interface Engine/Protocol Translator
- Software Development :: Libraries
- Software Development :: Libraries :: Python Modules

Quick Start

The Ticker module

The `Ticker` module, which allows you to access ticker data in a more Pythonic way:

```
import yfinance as yf

msft = yf.Ticker("MSFT")

# get all stock info
msft.info

# get historical market data
hist = msft.history(period="1mo")

# show meta information about the history (requires history() to be called fir
msft.history_metadata

# show actions (dividends, splits, capital gains)
msft.actions
msft.dividends
msft.splits
msft.capital_gains  # only for mutual funds & etfs

# show share count
msft.get_shares_full(start="2022-01-01", end=None)

# show financials:
# - income statement
msft.income_stmt
msft.quarterly_income_stmt
# - balance sheet
```

```
msft.balance_sheet
msft.quarterly_balance_sheet
# - cash flow statement
msft.cashflow
msft.quarterly_cashflow
# see `Ticker.get_income_stmt()` for more options

# show holders
msft.major_holders
msft.institutional_holders
msft.mutualfund_holders
msft.insider_transactions
msft.insider_purchases
msft.insider_roster_holders

# show recommendations
msft.recommendations
msft.recommendations_summary
msft.upgrades_downgrades

# Show future and historic earnings dates, returns at most next 4 quarters and
# Note: If more are needed use msft.get_earnings_dates(limit=XX) with increase
msft.earnings_dates

# show ISIN code - *experimental*
# ISIN = International Securities Identification Number
msft.isin

# show options expirations
msft.options

# show news
msft.news

# get option chain for specific expiration
```

```
opt = msft.option_chain('YYYY-MM-DD')
# data available via: opt.calls, opt.puts
```

If you want to use a proxy server for downloading data, use:

```
import yfinance as yf

msft = yf.Ticker("MSFT")

msft.history(..., proxy="PROXY_SERVER")
msft.get_actions(proxy="PROXY_SERVER")
msft.get_dividends(proxy="PROXY_SERVER")
msft.get_splits(proxy="PROXY_SERVER")
msft.get_capital_gains(proxy="PROXY_SERVER")
msft.get_balance_sheet(proxy="PROXY_SERVER")
msft.get_cashflow(proxy="PROXY_SERVER")
msft.option_chain(..., proxy="PROXY_SERVER")
...
```

Multiple tickers

To initialize multiple `Ticker` objects, use

```
import yfinance as yf

tickers = yf.Tickers('msft aapl goog')

# access each ticker using (example)
tickers.tickers['MSFT'].info
```

```
tickers.tickers['AAPL'].history(period="1mo")
tickers.tickers['GOOG'].actions
```

To download price history into one table:

```
import yfinance as yf
data = yf.download("SPY AAPL", period="1mo")
```

`yf.download()` and `Ticker.history()` have many options for configuring fetching and processing. Review the Wiki for more options and detail.

Logging

`yfinance` now uses the `logging` module to handle messages, default behaviour is only print errors. If debugging, use `yf.enable_debug_mode()` to switch logging to debug with custom formatting.

Smarter scraping

Install the `nospam` packages for smarter scraping using `pip` (see Installation). These packages help cache calls such that Yahoo is not spammed with requests.

To use a custom `requests` session, pass a `session=` argument to the Ticker constructor. This allows for caching calls to the API as well as a custom way to modify requests via the `User-agent` header.

```
import requests_cache
session = requests_cache.CachedSession('yfinance.cache')
session.headers['User-agent'] = 'my-program/1.0'
ticker = yf.Ticker('msft', session=session)
# The scraped response will be stored in the cache
ticker.actions
```

Combine `requests_cache` with rate-limiting to avoid triggering Yahoo's rate-limiter/blocker that can corrupt data.

```
from requests import Session
from requests_cache import CacheMixin, SQLiteCache
from requests_ratelimiter import LimiterMixin, MemoryQueueBucket
from pyrate_limiter import Duration, RequestRate, Limiter
class CachedLimiterSession(CacheMixin, LimiterMixin, Session):
    pass

session = CachedLimiterSession(
    limiter=Limiter(RequestRate(2, Duration.SECOND*5)),  # max 2 requests per 
    bucket_class=MemoryQueueBucket,
    backend=SQLiteCache("yfinance.cache"),
)
```

Managing Multi-Level Columns

The following answer on Stack Overflow is for How to deal with multi-level column names downloaded with yfinance?

- `yfinance` returns a `pandas.DataFrame` with multi-level column names, with a level for the ticker and a level for the stock price data

- The answer discusses:
    - How to correctly read the the multi-level columns after saving the dataframe to a csv with `pandas.DataFrame.to_csv`
    - How to download single or multiple tickers into a single dataframe with single level column names and a ticker column

`pandas_datareader` override

If your code uses `pandas_datareader` and you want to download data faster, you can "hijack" `pandas_datareader.data.get_data_yahoo()` method to use **yfinance** while making sure the returned data is in the same format as **pandas_datareader**'s `get_data_yahoo()`.

```
from pandas_datareader import data as pdr

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

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

Persistent cache store

To reduce Yahoo, yfinance store some data locally: timezones to localize dates, and cookie. Cache location is:

- Windows = C:/Users/<USER>/AppData/Local/py-yfinance
- Linux = /home/<USER>/.cache/py-yfinance
- MacOS = /Users/<USER>/Library/Caches/py-yfinance

You can direct cache to use a different location with `set_tz_cache_location()`:

```
import yfinance as yf
yf.set_tz_cache_location("custom/cache/location")
...
```

Developers: want to contribute?

`yfinance` relies on community to investigate bugs and contribute code. Developer guide: https://github.com/ranaroussi/yfinance/discussions/1084

Legal Stuff

yfinance is distributed under the **Apache Software License**. See the LICENSE.txt file in the release for details.

AGAIN - 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, here, and here) for detailes on your rights to use the actual data downloaded.

P.S.

Please drop me an note with any feedback you have.

Ran Aroussi

Site map

Switch to desktop version

English español français 日本語 português (Brasil) українська Ελληνικά Deutsch 中文 (简体) 中文 (繁體) русский עברית esperanto