pymarketcap Documentation

Release 4.0.0017

Álvaro Mondéjar Rubio

Oct 29, 2018

User's guide

CHAPTER 1

Install

1.1 From Pypi

You need to install cython before pymarketcap. Try: `pip3 install Cython` and then:

```
pip3 install pymarketcap
```

- On Windows will be used `urllib` library to make synchronous requests and on Linux/Mac will be build against libcurl C library. You can control this (see below):

1.2 From source

```
git clone https://github.com/mondeja/pymarketcap.git
cd pymarketcap
pip3 install -r requirements.txt
python setup.py install
```

- To force installation with libcurl, use `--force-curl` in last command.
- To install with urllib, use `--no-curl`.

1.3 Known install issues

```
pymarketcap/core.c:16:20: fatal error: Python.h: No such file or directory
```

- Solution (Linux): `sudo apt-get install python3-dev`

```
pymarketcap/curl.c:581:23: fatal error: curl/curl.h: No such file or directory
```

- Solution (Linux): `sudo apt-get install libcurl4-openssl-dev`

CHAPTER 2

Usage

Check out complete live demos hosted at Binderhub.

2.1 Basic examples

Synchronous Interface

```
from pymarketcap import Pymarketcap
cmc = Pymarketcap()

cmc.exchanges()
```

Asynchronous Scraper

```
import asyncio
from pymarketcap import AsyncPymarketcap

async def main():
    async with AsyncPymarketcap() as apym:
        async for currency in apym.every_currency():
            print(currency)

if __name__ == '__main__':
    loop = asyncio.get_event_loop()
    loop.run_until_complete(main())
```

CHAPTER 3

Reference

3.1 pymarketcap.core

class pymarketcap.core.**Pymarketcap**

Synchronous class for retrieve data from https://coinmarketcap.com.

Parameters

- **timeout** (*int, optional*) – Set timeout value for requests. As default `20`.
- **debug** – (bool, optional): Show low level data in get requests. As default, `False`.
- **proxy_addr** (*bytes, optional*) – Proxy to use with Pymarketcap. As default, `b""`.

Coinmarketcap API

listings()

List all criptocurrencies with their ids, names, symbol and website slug.

Returns (dict): Coinmarketcap API raw response.

stats()

Get global cryptocurrencies statistics.

Parameters **convert** (*str, optional*) – return 24h volume, and market cap in terms of another currency. See `ticker_badges` property to get valid values. As default `"USD"`.

Returns (dict): Global markets statistics on a raw response.

ticker()

Get currencies with other aditional data. Only returns 100 currencies in each request. Use `pymarketcap.Pymarketcap.ticker_all()` method for retrieve all currencies navegation through API pagination.

Parameters

- **currency** (*str, optional*) – Specify a currency to return data, that can be a `name`, `symbol`, `id` or `website_slug` fields from *pymarketcap.core.Pymarketcap.cryptocurrencies* property. If you dont specify a currency, returns data for all in coinmarketcap. As default, `None`.
- **convert** (*str, optional*) – Allows to convert prices, 24h volumes and market capitalizations in terms of one of badges returned by `ticker_badges` property. As default, `"USD"`.

Returns (dict): Data from all currencies or a currency from coinmarketcap.

cryptocurrencies
Return all cryptocurrencies listed at coinmarketcap. This is the cached version of public API listings method but without low level fields like `"data"` and `"metadata"`.

ticker_badges
Badges in which you can convert prices in `ticker()` method.

Web scraper

currency()
Get currency metadata like total markets capitalization, websites, source code link, if mineable...

Parameters

- **curr** (*str*) – Currency to get metadata. Can be a `name`, `symbol`, `id` or `website_slug` fields from *pymarketcap.core.Pymarketcap.cryptocurrencies*.
- **convert** (*str, optional*) – Currency to convert response fields `total_markets_cap`, `total_markets_volume_24h` and `price` between USD and BTC. As default `"USD"`.

Returns (dict): Aditional general metadata not supported by other methods.

exchange()
Obtain data from a exchange passed as argument. See *pymarketcap.core.Pymarketcap.cryptoexchanges* property for obtain all posibles values.

Parameters

- **exc** (*str*) – Exchange to retrieve data. Can be a `name`, `id` or `website_slug` fields from *pymarketcap.core.Pymarketcap.cryptoexchanges*.
- **convert** (*str, optional*) – Convert prices and 24h volumes in return between USD and BTC. As default `"USD"`.

Returns (dict): Data from a exchange. Fields: `"currency"`, `"pair"`, `"name"`, `"volume_24h"` (total), `"price"`, `"percent_volume"`, `"updated"`. `"slug"`, `"website_slug"`, `"id"`, `"volume"`, `"markets"`.

exchanges()
Get all exchanges in coinmarketcap ranked by volumes along with other metadata.

Parameters **convert** (*str, optional*) – Convert volumes and prices between USD and BTC. As default `"USD"`.

Returns (list): Exchanges with markets and other data included.

cryptoexchanges
Returns all exchanges listed at coinmarketcap, as dictionaries with "name", "id" and "website_slug" keys.

historical()
Get historical data for a currency.

Parameters

- **curr** (*str*) – Currency to scrap historical data. Can be a name, "symbol", id or website_slug fields from *pymarketcap.core.Pymarketcap.cryptocurrencies*.
- **start** (*date, optional*) – Time to start scraping periods as datetime.datetime type. As default datetime.datetime(2008, 8, 18)().
- **end** (*date, optional*) – Time to end scraping periods as datetime.datetime type. As default datetime.datetime.now().
- **revert** (*bool, optional*) – If False, return first date first, in chronological order, otherwise returns reversed list of periods. As default False.

Returns (list): Historical dayly OHLC for a currency.

markets()
Get available coinmarketcap markets data for a currency.

Parameters

- **curr** (*str*) – Currency to get market data. Can be a name, "symbol", id or website_slug fields from *pymarketcap.core.Pymarketcap.cryptocurrencies*.
- **convert** (*str, optional*) – Currency to convert response fields volume_24h and price between USD and BTC. As default "USD".

Returns (list): Markets on wich provided currency is currently tradeable.

ranks()
Returns gainers and losers for 1 hour, 24 hours and 7 days.

Returns (dict): A dictionary with 2 keys (gainers and losers) whose values are the periods "1h", "24h" and "7d".

recently()
Get recently added currencies along with other metadata.

Parameters **convert** (*str, optional*) – Convert market_caps, prices, volumes and percent_changes between USD and BTC. As default "USD".

Returns (list): Recently added currencies data.

tokens()
Get data from platforms tokens

Parameters **convert** (*str, optional*) – Convert "market_cap", "price" and "volume_24h" values between USD and BTC. As default "USD".

Returns (list): Platforms tokens data.

Graphs API

Note: The `graphs` methods can be called also as `cmc.graphs.currency()`, `cmc.graphs.global_cap()` and `cmc.graphs.dominance()`, being `cmc` a instance of `Pymarketcap` or `AyncPymarketcap` classes.

_currency()
Get graphs data of a currency.

Parameters

- **curr** (*str*) – Currency to retrieve graphs data.
- **start** (*datetime, optional*) – Time to start retrieving graphs data in datetime type. As default `None`.
- **end** (*datetime, optional*) – Time to end retrieving graphs data in datetime type. As default `None`.
- **use_auto_timeframe** (*bool, optional*) – Use auto time frames same as fronted API. As default `False`

Returns (dict): Dict info with next keys: `"market_cap_by_available_supply"`, `"price_btc"`, `"price_usd"`, `"volume_usd":` and `"price_platform"`. For each value, a list of lists where each one has two values [<datetime>, <value>]

_global_cap()
Get global market capitalization graphs, including or excluding Bitcoin.

Parameters

- **bitcoin** (*bool, optional*) – Indicates if Bitcoin will be includedin global market capitalization graph. As default `True`.
- **start** (*int, optional*) – Time to start retrieving graphs data in datetime. As default `None`.
- **end** (*optional, datetime*) – Time to start retrieving graphs data in datetime. As default `None`.
- **use_auto_timeframe** (*bool, optional*) – Use auto time frames same as fronted API. As default `False`

Returns (dict): Whose values are lists of lists with timestamp and values, a data structure with the keys: `"volume_usd"` and `"market_cap_by_available_supply"`.

_dominance()
Get currencies dominance percentage graph

Parameters

- **start** (*int, optional*) – Time to start retrieving graphs data in datetime. As default `None`.
- **end** (*optional, datetime*) – Time to start retrieving graphs data in datetime. As default `None`.
- **use_auto_timeframe** (*bool, optional*) – Use auto time frames same as fronted API. As default `False`

Returns (dict): Altcoins and dominance percentage values with timestamps.

Utils

convert()

Convert prices between currencies. Provide a value, the currency of the value and the currency to convert it and get the value in currency converted rate. For see all available currencies to convert see `currencies_to_convert` property.

Parameters

- **value** (*int/float*) – Value to convert betweeen two currencies.
- **currency_in** (*str*) – Currency in which is expressed the value passed.
- **currency_out** (*str*) – Currency to convert.

Returns (float): Value expressed in currency_out parameter provided.

download_logo()

Download a currency image logo providing their size.

Parameters

- **curr** (*str*) – Currency name, id, website_slug or symbol to download.
- **size** (*int, optional*) – Size in pixels. Valid sizes are: `[16, 32, 64, 128, 200]`. As default `128`.
- **filename** (*str, optional*) – Filename for store the logo. Doesn't include the extension (will be ".png"). As default `None`.

Returns (str): Filename of downloaded file if all was correct.

download_exchange_logo()

Download a exchange logo passing his name or id or website slug as first parameter and optionally a filename without extension.

Parameters

- **exc** (*str*) – Exchange name, id or website slug to download.
- **size** (*int*) – Size in pixels. Valid values are: `[16, 32, 64, 128, 200]`.
- **filename** (*str, optional*) – Filename for store the logo, without include file extension (will be ".png"). As default `None`.

Returns (str): Filename of downloaded file if all was correct.

3.2 pymarketcap.pymasyncore

class pymarketcap.pymasyncore.**AsyncPymarketcap**(*queue_size=10, progress_bar=True, consumers=10, timeout=15, logger=<Logger /pymarketcap/pymasyncore.py (WARNING)>, debug=False, sync=<pymarketcap.core.Pymarketcap object>, **kwargs*)

Bases: `aiohttp.client.ClientSession`

Asynchronous scraper for coinmarketcap.com

Parameters

- **queue_size** (*int*) – Number of maximum simultanenous get requests performing together in methods involving several requests. As default `10`.
- **progress_bar** (*bool*) – Select `True` or `False` if you want to show a progress bar in methods that involve processing of several requests (requires `tqdm` module). As default, `True`.
- **consumers** (*int*) – Number of consumers resolving HTTP requests from an internal `asyncio.Queue`. As default, `10`.
- **timeout** (*int/float, optional*) – Limit max time waiting for a response. As default, `15`.
- **logger** (*logging.logger*) – As default is a logger with a `StreamHandler`.
- **debug** (*bool, optional*) – If `True`, the logger level will be setted as `DEBUG`. As default `False`.
- **sync** (*object, optional*) – Synchronous version instance of pymarketcap. As default *pymarketcap.core.Pymarketcap*
- ****kwargs** – arguments that corresponds to the `aiohttp.client.ClientSession` parent class.

Note: All scraper methods described in `Pymarketcap` object and almost all the properties also exists in `AsyncPymarketcap`.

---

every_currency(*currencies=None*, *convert='USD'*, *consumers=None*)

Return general data from every currency in coinmarketcap passing a list of currencies as first parameter. As default returns data for all currencies.

Parameters

- **currencies** (*list, optional*) – Iterator with all the currencies that you want to retrieve. As default `None` (`pymarketcap.Pymarketcap.coins()` will be used in that case).
- **convert** (*str, optional*) – Convert prices in response between "USD" and BTC. As default `"USD"`.
- **consumers** (*int, optional*) – Number of consumers processing the requests simultaneously. As default `None` (see `pymarketcap.AsyncPymarketcap.consumers`).

Returns (list): Data for all currencies.

every_markets(*currencies=None*, *convert='USD'*, *consumers=None*)

Returns markets data from every currency in coinmarketcap passing a list of currencies as first parameter. As default returns data for all currencies.

Parameters

- **currencies** (*list, optional*) – Iterator with all the currencies that you want to retrieve. As default `None` (`pymarketcap.Pymarketcap.coins()` will be used in that case).
- **convert** (*str, optional*) – Convert prices in response between "USD" and BTC. As default `"USD"`.

- **consumers** (*int, optional*) – Number of consumers processing the requests simultaneously. As default None (see pymarketcap.AsyncPymarketcap.consumers).

Returns (async iterator): Data for all currencies.

every_historical(*currencies=None*, *start=datetime.datetime(2008, 8, 18, 0, 0)*, *end=datetime.datetime(2018, 10, 29, 21, 13, 35, 884743)*, *revert=False*, *consumers=None*)

Returns historical data from every currency in coinmarketcap passing a list of currencies as first parameter. As default returns data for all currencies.

Parameters

- **currencies** (*list, optional*) – Iterator with all the currencies that you want to retrieve. As default None (pymarketcap.Pymarketcap.coins() will be used in that case).
- **start** (*date, optional*) – Time to start scraping periods as datetime.datetime type. As default datetime(2008, 8, 18).
- **end** (*date, optional*) – Time to end scraping periods as datetime.datetime type. As default datetime.now().
- **revert** (*bool, optional*) – If False, return first date first, in chronological order, otherwise returns reversed list of periods. As default False.
- **consumers** (*int, optional*) – Number of consumers processing the requests simultaneously. As default None (see pymarketcap.AsyncPymarketcap.consumers).

Returns (async iterator): Historical data for all currencies.

every_exchange(*exchanges=None*, *convert='USD'*, *consumers=None*)

Returns general data from every exchange in coinmarketcap passing a list of exchanges as first parameter. As default returns data for all exchanges.

Parameters

- **exchanges** (*list, optional*) – Iterator with all the exchanges that you want to retrieve. As default None (pymarketcap.Pymarketcap.exchange_slugs() will be used in that case).
- **convert** (*str, optional*) – Convert market_caps, prices, volumes and percent_changes between USD and BTC. As default "USD".
- **consumers** (*int, optional*) – Number of consumers processing the requests simultaneously. As default None (see pymarketcap.AsyncPymarketcap.consumers).

Returns (async iterator): General data from all exchanges.

Note: The next method can be called also as graphs.every_currency().

_every_currency(*currencies=None*, *start=None*, *end=None*, *use_auto_timeframe=False*, *consumers=None*)

Returns graphs data from every currency in coinmarketcap passing a list of currencies as first parameter. As default returns graphs data for all currencies.

Parameters

- **currencies** (*list, optional*) – Iterator with all the currencies that you want to retrieve. As default `None` (`pymarketcap.Pymarketcap.coins()` will be used in that case).
- **start** (*datetime, optional*) – Time to start retrieving graphs data in datetime. As default `None`.
- **end** (*datetime, optional*) – Time to end retrieving graphs data in datetime. As default `None`.
- **use_auto_timeframe** (*bool, optional*) – Use auto time frames same as fronted API. As default `False`
- **consumers** (*int, optional*) – Number of consumers processing the requests simultaneously. As default `None` (see `pymarketcap.AsyncPymarketcap.consumers`).

Returns (async iterator): Graphs data from all currencies.

3.3 pymarketcap.errors

Pymarketcap errors module.

exception pymarketcap.errors.**CoinmarketcapError**
Bases: `Exception`

Coinmarketcap base classs errors.

exception pymarketcap.errors.**CoinmarketcapHTTPError**
Bases: *pymarketcap.errors.CoinmarketcapError*

Exception for catch HTTPErrors.

exception pymarketcap.errors.**CoinmarketcapHTTPError404**
Bases: *pymarketcap.errors.CoinmarketcapError*

Exception for catch 404 HTTP error codes.

exception pymarketcap.errors.**CoinmarketcapHTTPError408**
Bases: *pymarketcap.errors.CoinmarketcapError*

Exception for catch request timeout HTTP errors.

exception pymarketcap.errors.**CoinmarketcapTooManyRequestsError**
Bases: *pymarketcap.errors.CoinmarketcapHTTPError*

Exception for catch 429 HTTP error codes.

CHAPTER 4

Contributing

4.1 Known issues and enhacements

- Total:
- Tests:

4.2 Basic guidelines

- Each new method developed needs to be accompanied with their respective complete unittest as shown in `tests/` directory.
- Each new pull request needs to be good performed. Plase, don't make a pull request with 4 commits for change a line in the code.

4.3 Basic benchmarking

You can test basically benchmarking of `Pymarketcap` class methods running `python3 bench/main.py`. You can filter by name of benchmarks, change the number of repetitions for each one or change file where benchmarks results are stored: run `python3 bench/main.py --help`.

4.4 How does pymarketcap works in depth?

- Some pieces of code are precompiled before compile with Cython, so if you see missing parts on the source code before install (like the property method `ticker_badges`), understand that these are not bugs. Run `make precompile-sources` to do manual code precompilation and `make restore-sources` for restore souce code to original state.
- Numerical values returned by the scraper are real values with which coinmarketcap.com works, not the values displayed on their frontend (see source HTML code of the web).

4.5 Contributors

4.5.1 Pull requesters

- badele
- nkoshell
- wilcollins

4.5.2 Bug hunters

- Abolah
- badele
- Bragegs
- criptonaut1357
- Prophetic-Pariah
- reteps
- run2dev
- Wingie

CHAPTER 5

Testing

You need to install `pytest` for run unittests:

```
pip3 install -r dev-requirements.txt
```

You can run tests with `pytest` command:

- Run all unittests: `pytest tests`
- Run also tests for asynchronous interface: `pytest tests --end2end`
- Run individual tests:
  - Run API tests: `pytest tests/test_sync_core/test_public_api`
  - Run `every_historical()` async scraper method's consistence: `pytest tests/test_async_core/test_scraper/test_every_historical.py`

Also, if your system is Unix, you can use `make` for run tests, install, precompile/restore source code, build and clean the whole directory (see Makefile).

You can see online tests for Linux/Mac and Windows based systems at TravisCI and AppVeyor:

- TravisCI
- AppVeyor

CHAPTER 6

Changelog

6.1 4.1.0 (in process)

- Some tests added which assert if a field is `None` between all fields methods responses (see #46). These check if a field are not being parsed by `processer.pyx` regular expressions.
- New method `ticker_all` due to coinmarketcap API has implemented a limit of 100 for the number of currencies in `/ticker/` endpoint responses. With `ticker_all` we can retrieve all currencies from `ticker` method responses.
- New parameter added `use_auto_timeframe` in some methods until pull request #52.

6.2 4.0.0

- Coinmarketcap API is updated to version 2 providing the listing endpoint in their API. Also, parameters like `usd_market_cap` have been missing and instead a `quotes` field list prices by currencies in responses.
- Some methods of `Pymarketcap` class have been deprecated: `correspondences`, `ids_correspondences`, `_is_symbol`, `_cache_symbols_ids`, `_cache_exchanges_ids`, `symbols`, `coins`, `total_currencies`, `exchange_names`, `__exchange_names_slugs`, `exchange_slugs` and `total_exchanges`.
- Next methods have been added instead: `cryptocurrencies`, `cryptocurrency_by_field_value`, `cryptoexchanges`, `exchange_by_field_value`, `field_type` and `listings`. All methods listed in previous point have been replaced by these 6 methods, simplifying the process of access to cryptocurrencies and exchanges and allowing to retrieve them through any field: `name`, `symbol`, `website_slug` and `id` (cryptocurrencies) or `name`, `website_slug` and `id` (exchanges).
- New method `download_exchange_logo` for synchronous interface.

6.3 3.9.0

- All the wrapper rewritten with Cython language.
- The data is obtained and processed by regular expressions instead of parsing the DOM tree.
- Core functionality of the wrapper rewritten for work with `libcurl` C library through a Cython wrap at compilation time. Also, you can use the wrapper with `urllib` standard library only installing by `python setup.py install --no-curl`.
- Tests now performed with `pytest` instead of standard library `unittest`.
- `request`, `lxml` and `bs4` dependencies removed, only `cython`, `gcc` and `libcurl` required for compile the code.
- A precompiler added for insert some code and documentation hardcoded before compile the program.
- All the data now is taken from values provided for the code that builds coinmarketcap instead the values displayed in the frontend page, as before. Is possible select between USD or BTC to returns these in most methods.
- New method `convert()` for convert between currencies as coinmarketcap currencies calculator: https://coinmarketcap.com/calculator/
- New method `tokens()` convering partially https://coinmarketcap.com/tokens/views/all/ endpoint.
- New method `currency()` for get all metadata from a currency.
- New asynchronous class interface with methods for retrieve faster long lists of exchanges or currencies: `every_currency()`, `every_exchange()`, `every_historical()`.
- Improvements in both speed and accuracy in exchanges and currencies cache, from `quick_search.json` and `quick_search_exchanges.json` files of coinmarketcap server.

6.4 3.3.0

- New method `download_logo()` that downloads images for all coins in coinmarketcap in various sizes.
- New methods for retrieve info from `graphs` coinmarketcap internal API: `graphs.currency`, `graphs.global_cap` and `graphs.dominance`
- Some symbols recognition improvements and bugs fixed.

This proyect is originally a fork from https://github.com/barnumbirr/coinmarketcap

Python Module Index

p

Index

Symbols

A

C

D

E

H

L

M

P

R

S

T