API Adapter

Code available at: https://git.fhict.nl/I404788/trading-bot/-/blob/master/tradeEnv/api_adapter.py
Example available at: https://git.fhict.nl/I404788/trading-bot/-/blob/master/test_binance.py

This module provides a almost universal adapter for scraping RESTful APIs, this is entirely data-oriented.

The only requirement is to create a 'map' of bindings for which endpoints receive what kind of information. This is helpful when adapting our code for different exchange as it would only require minimal changes.

Assumptions:

- Data received is always in a semi-structured jagged-array/dictionary
- Cursor based APIs have an limit+offset, or a recursive progression
- The API has headers or error codes for rate-limiting
- All parameters can be encoded in the URL
  - This might be upgraded later on, but `Binance` our main exchange only requires url encoded.

Examples

For examples see the bottom of `api_adapter.py`

Specifications

API Adapter specifies a DSL-ish way to convert many different APIs to a common format.

To ensure the operations are executed in the correct order all endpoint maps must be `OrderedDict`s.

On the left side:

- if a path ends with `|*|`; it's an item the Set
- if a path ends with `{*}`; it's the ID of that item
- if a path contains `[*]`; it's an item the array/object
- if a path contains `{:SomeID}`; it's an internal variable of the adapter
- if a path contains `{SomeID}`; it's an parameter to be filled in by the user

The right side follow the jsonpath specifications (with extensions).

Predefined

There are some predefined rules for convience:

- `{:cursor}` indicates a cursorable endpoint, in combination with `{:limit}` this can be used to scrape large amounts of data
  - If no `{:limit}` is used, the adapter will try to fallback onto the data it has received for the next cursor. This can be used when an the cursor are not indexes, but for example a timestamp. Do note that this is very experimental.