cad$_{t}ickers$

Release v0.1.3

Aug 23, 2020

Contents:

CHAPTER 1

Canadian Securities Exchange

Functions to download tickers from the cse

`cad_tickers.exchanges.cse.`**`add_descriptions_to_df`**(*df: pandas.core.frame.DataFrame*, *max_workers: int = 16*) → pandas.core.frame.DataFrame

Parameters:

clean_df Dataframe with with randomly selected values. Data columns are as follows:

Company	Full name of the company
Symbol	Listing symbol from the cse exchange needs a mapper to yahoo finance
Industry	Enum of industry including Mining
Identifier	Broad category (US Cannabis)
Indices	Enum such as CSE Composite
Currency	Usually CAD
Trading	Date when trading started
urls	url to listing on cse website

max_workers maximum number of thread workers to have

Returns:

df Dataframe descriptions in every column if valid

Company	Full name of the company
Symbol	Listing symbol from the cse exchange needs a mapper to yahoo finance
Industry	Enum of industry including Mining
Identifier	Broad category (US Cannabis)
Indices	Enum such as CSE Composite
Currency	Usually CAD
Trading	Date when trading started
urls	url to listing on cse website
description	cse description scrapped from website

`cad_tickers.exchanges.cse.`**`clean_cse_data`**(*raw_df: pandas.core.frame.DataFrame*) → pandas.core.frame.DataFrame

Removes bad data from cse dataframe.

Parameters:

raw_df Dataframe with mostly unnamed columns from pandas df import

CSE Listings	Label for Company data
Unnamed: 1	Listing symbol from the cse exchange needs a mapper to yahoo finance
Unnamed: 2	Enum of industry including Mining
Unnamed: 3	Enum such as CSE Composite
Unnamed: 4	Enum such as CSE Composite
Unnamed: 5	Usually CAD
Unnamed: 6	empty (pandas import error, dropped)
Unnamed: 7	Date when trading started

Returns:

clean_df Dataframe with bad data removed

Company	Full name of the company
Symbol	Listing symbol from the cse exchange needs a mapper to yahoo finance
Industry	Enum of industry including Mining
Identifier	Broad category (US Cannabis)
Indices	Enum such as CSE Composite
Currency	Usually CAD
Trading	Date when trading started
urls	url to listing on cse website

`cad_tickers.exchanges.cse.`**`get_cse_files`**(*filename: str = 'cse.xlsx', filetype: str = 'xlsx'*) → str

Gets excel spreadsheet from api.tsx using requests

Parameters: filename: Name of the file to be saved filetype: Save as pdf or xlsx

Returns: filePath returns path to file

See ://stackoverflow.com/questions/13567507/passing-csrftoken-with-python-requests

`cad_tickers.exchanges.cse.`**`get_cse_tickers_df`**() → pandas.core.frame.DataFrame

Grab cse dataframe from exported xlsx sheet

Returns:

clean_df Dataframe with with randomly selected values. Data columns are as follows:

Company	Full name of the company
Symbol	Listing symbol from the cse exchange needs a mapper to yahoo finance
Industry	Enum of industry including Mining
Identifier	Broad category (US Cannabis)
Indices	Enum such as CSE Composite
Currency	Usually CAD
Trading	Date when trading started
urls	url to listing on cse website

`cad_tickers.exchanges.cse.`**`get_description_for_url`**(*url: str*) → str

Parameters: url - link to ticker can be empty string

Returns: description - details of what the ticker does, can be empty string

CHAPTER 2

Toronto Stock Exchange

Set of functions to scrap ticker data from the toronto stock exchange.

Will definitely split into smaller files once the graphql api becomes the main api.

`cad_tickers.exchanges.tsx.`**`add_descriptions_to_df`**(*df*) → pandas.core.frame.DataFrame

Description: single process solution to fetching descriptions

Input: df: dataframe containing tickers

Returns: df: updated dataframe with a descriptions if available

`cad_tickers.exchanges.tsx.`**`add_descriptions_to_df_pp`**(*df: pandas.core.frame.DataFrame*, *max_workers: int = 16*) → pandas.core.frame.DataFrame

Description: fetch descriptions for tickers in parallel noticable speedup uses thread pool which should be faster

Input: df: dataframe containing tickers

Returns: df: updated dataframe with a descriptions if available

`cad_tickers.exchanges.tsx.`**`add_descriptions_to_df_pp_legacy`**(*df: pandas.core.frame.DataFrame*) → pandas.core.frame.DataFrame

Description: fetch descriptions for tickers in parallel noticable speedup, keeping this to verify speed increase

Input: df: dataframe containing tickers

Returns: df: updated dataframe with a descriptions if available

`cad_tickers.exchanges.tsx.`**`company_description_by_ticker`**(*ticker*) → str

Description: Grabs searchable ticker from quotemedia using tmx ticker

Input: ticker: string

Returns: df: updated dataframe with a descriptions if available

`cad_tickers.exchanges.tsx.`**`dl_tsx_xlsx`**(*filename: str = ''*, ***kwargs*) → str

Description: Gets excel spreadsheet from the tsx api using programatically

Note: Replicates api calls in TSX discover tool with all parameters. See migreport search Note that not all parameters are documented and/or limited validation

Parameters: filename: Name of the file to be saved

Kwargs:

- exchanges (string): TSX, TSXV
- marketcap (string): values from 0 to specified value
- sectors (string): cpc, clean-technology, closed-end-funds, technology

Returns: data - returns path to file or pandas dataframe

pd.DataFrame

Ex.	Exchange ticker in TSXV, TSX
Name	Full name of ticker
Ticker	Symbol usually 4 characters or less
QMV($)	Quoted Market Value, I assume this is based on the "currency".
HQ Region	Headquarters region usually a country (need to double check)
HQ Location	Usually a province or state
Sector	Main sector, technology
Sub Sector	Sub Sector

See passing csrftoken

`cad_tickers.exchanges.tsx.`**`get_description_for_ticker`**(*ticker: str*) → str
set of functionality

`cad_tickers.exchanges.tsx.`**`get_mig_report`**(*filename: str = ", exchange: str = 'TSX', return_df: bool = False*) → str

Description: Gets excel spreadsheet from tsx api programatically. See for more flexibility `dl_tsx_xlsx`

Parameters: filename: Name of the file to be saved exchanges: TSX, TSXV return_df: Return a pandas dataframe

Returns: filePath: returns path to file or dataframe

See ://stackoverflow.com/questions/13567507/passing-csrftoken-with-python-requests

`cad_tickers.exchanges.tsx.`**`grab_symbol_for_ticker`**(*ticker: str*) → str

Description: Grabs the first symbol from ticker data all symbols should lead to valid webpages for data scrapping.

Parameters: ticker: string representing the stock ticker

Returns: symbol: string - searchable string in the quotemedia api or empty string

`cad_tickers.exchanges.tsx.`**`lookup_symbol_by_ticker`**(*ticker: str*) → list
Description: Returns search array dictionary for tickers

Note: sometimes the name of the ticker in the xlsx sheet is off slightly and we need to find the "real ticker". Uses standard api (not graphql) to grab tickers

Example searchpoint is https://app.quotemedia.com/lookup?callback=tmxtickers&q=zmd&limit=5&webmasterId=101020

See Tmx Graphql and the new tmx site

Input: ticker: tmx ticker

Output: quote_data: list of ticker metadata

CHAPTER 3

Stock News

Extract news from stocks on yahoo

`cad_tickers.news.stock_news.`**`find_news_link_and_text`**(*news_content: bs4.element.Tag*) → Tuple[str, str]

Finds news link from news_content. Assumes comments are deleted from the yahoo finance news items

Parameters: news_content - html based data for the news article

Returns: link_href - link in html markup link_text - link text in html markup

`cad_tickers.news.stock_news.`**`find_news_source`**(*news_content: bs4.element.Tag*) → Union[None, str]

Utility function to verify news format from yahoo has not changed

when grabbing data from yahoo with requests, it seems date is not returned.

Parameters: news_content: html based data for the news article

Returns: source - publisher of article

wrapper div around content - such as - CNW Group 2 days ago

`cad_tickers.news.stock_news.`**`get_ynews_for_ticker`**(*ticker: str, yahoo_base_url='https://finance.yahoo.com'*) → List[bs4.element.Tag]

Returns initial news items fetched from yahoo when loading quote page. Since yahoo has lazy loading, not all items are returned. Seems like ads are not loaded because of lazy loading.

Parameters: ticker - yahoo formatted ticker str yahoo_base_url - optional parameter that is the base of the request

Returns: news_items - list of key html content for the news item

`cad_tickers.news.stock_news.`**`scrap_news_for_ticker`**(*ticker: str*) → List[dict]

Extracts webpage data from a ticker

Parameters: ticker - yahoo finance ticker

Returns:

news_data - list of dicts extracted from webpage

- source - str
- link_href - link from post (can be relative or absolute)
- link_text - description for link
- ticker - reference to original ticker

CHAPTER 4

IIROC Halts

Find out what latest stocks have been halted from iiroc (only canada)

`cad_tickers.news.iiroc_halts.`**`get_halts_resumption`**`()` → pandas.core.frame.DataFrame

Gets the latest 25 halts from the iiroc

Returns:

halt_df Dataframe with bad data removed

Halts	Details of halts
Listing	Extracted ticker from halt

CHAPTER 5

Stock Utilities

Contains various utility classes

`cad_tickers.util.utils.`**`convert`**`(`*file_path: str*`)` → pandas.core.frame.DataFrame

Parameters: file_path: path to excel sheet

`cad_tickers.util.utils.`**`make_cse_path`**`(`*raw_ticker: str*, *raw_industry: str*`)` → str
makes slug for ticker for the cse

Parameters: raw_ticker: cse ticker from xlsx sheet raw_industry: verbatim industry from ticker, not slugified

Returns: description: url for cse files for download

`cad_tickers.util.utils.`**`parse_description_tags`**`(`*description_tags: List[bs4.element.Tag]*`)` → str

Parameters: description_tags: html tags from webpage, usually p tag containing description

Returns: description: description for ticker

`cad_tickers.util.utils.`**`transform_name_to_slug`**`(`*raw_ticker: str*`)`

Parameters: raw_ticker: cse ticker to be converted to slug

CHAPTER 6

Examples

Grab Descriptions for all tsx tickers

```
from cad_tickers.exchanges.tsx import dl_tsx_xlsx, add_descriptions_to_df_pp
from datetime import datetime
start_time = datetime.now()
df = dl_tsx_xlsx()
# df = add_descriptions_to_df(df)
df = add_descriptions_to_df_pp(df)
end_time = datetime.now()
df.to_csv('tsx_all_descriptions.csv')
print(end_time - start_time)
```

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Index