investpy Documentation

Release 1.0

Alvaro Bartolome

Dec 25, 2020

Contents:

CHAPTER 1

Introduction

investpy is a Python package developed in order to retrieve all the available historical data from stocks/stocks, funds and ETFs from Investing.com. As Investing.com does not have any API to retrieve historical data, the main goal of this package is to allow users retrieve information from all the available financial products.

investpy came to life due to the need of covering the existing shortcomings in terms of real time data retrieval from stocks of the companies that make up the Spanish Stock Market, until the date there was no other package that provided a data extraction model for stocks from the Spanish Stock Market.

As time passed by, a decision was made on how investpy could be improved, and as the package was expected to have a high scalability and thus cover all the data possibilities offered by Investing to the public, investpy is now trying to expand the data it retrieves to make it more useful.

Along this document some relevant features of investpy are going to be sorted out and its functions are going to be explained in order to clarify its use.

1.1 Getting Started

Note: In order to get started using investpy you will need to have it installed, so if you do not have it already, check *Installation*.

Once you have investpy installed, you can now proceed to use the package. The first step is importing it at the top of your Python file as:

```
import investpy
```

Currently the main functions of investpy support historical data retrieval of stocks, funds and ETFs from all around the world (as indexed in Investing.com). Additionally to historical data retrieval, investpy also offers additional data retrieval related to the indexed financial products.

In order to clarify this concepts, some investpy functions are going to be presented, even though all of them are going to be properly explained and sorted out on their respective appendix in the documentation or in the API Reference.

For example, a block of code in order to get to test investpy usage is presented:

```
import investpy


# Retrieve all the available stocks as a Python list
stocks = investpy.get_stocks_list()

# Retrieve the recent historical data (past month) of a stock as a pandas.DataFrame␣
↪on ascending date order
df = investpy.get_stock_recent_data(stock='bbva', country='spain', as_json=False,␣
↪order='ascending')

# Retrieve the company profile of the introduced stock on english
profile = investpy.get_stock_company_profile(stock='bbva', country='spain', language=
↪'english')
```

1.2 Data Source

Investing is the main data source from which investpy retrieves the data. Investing is a global financial portal and Internet brand owned by Fusion Media Ltd. which provides news, analysis, streaming quotes, charts, technical data and financial tools about the global financial markets.

So as, the decision of choosing Investing as the data source is based on its reliability and also because it is one of the few web pages that provide detailed data from spanish markets, as it was the main focus when determining to develop the package as explained previously.

CHAPTER 2

Installation

Note: After installing the package you are now available to use it! As investpy's latest release is 0.9.8 the installation is optimized for it. If you try installing another investpy release, some features may not work.

2.1 First Installation

In order to get this package working you will need to install it on its last version. To install the package on either way you will need to have a Python 3.x version installed and pip or conda, so you can install Python packages from PyPI and from Anaconda Cloud, respectively. So, to install the latest release of investpy, you can either do it:

- via Python Package Indexer (PyPI):

```
$ python -m pip install investpy
```

- via Anaconda Cloud:

```
$ conda install investpy
```

- from GitHub via PyPI:

```
$ python -m pip install https://github.com/alvarobartt/investpy/archive/master.zip
```

2.2 Update Package

If you already had investpy installed and you want to update it you can do it:

- via PyPI:

```
$ python -m pip install --upgrade investpy
```

- via Anaconda Cloud:

```
$ conda update investpy
```

- from GitHub via PyPi:

```
$ python -m pip install --upgrade https://github.com/alvarobartt/investpy/archive/
↪master.zip
```

All the dependencies are already listed on the setup file of the package, but to sum them up, when installing investpy, it will install the following dependencies:

- pandas 0.25.1
- requests 2.22.0
- lxml 4.4.1
- unidecode 1.1.1

CHAPTER 3

Usage

Along this document, the main investpy functions are going to be presented. So on, this is a tutorial on how to use **investpy** to retrieve data from the financial products available, such as: stocks, funds, ETFs, indices and currency crosses, retrieved from Investing.com.

3.1 Recent/Historical Data Retrieval

The main functionallity of **investpy** is to retrieve historical data from the indexed financial products. So both recent and historical data retrieval functions have been developed in order to retrieve data from the last month or from a concrete period of time, respectively.

3.1.1 Stock Data Retrieval

```
import investpy

df = investpy.get_stock_recent_data(stock='bbva',
                                    country='spain')
print(df.head())

              Open   High    Low  Close    Volume Currency
Date
2019-08-13  4.263  4.395  4.230  4.353  27250000      EUR
2019-08-14  4.322  4.325  4.215  4.244  36890000      EUR
2019-08-15  4.281  4.298  4.187  4.234  21340000      EUR
2019-08-16  4.234  4.375  4.208  4.365  46080000      EUR
2019-08-19  4.396  4.425  4.269  4.269  18950000      EUR
```

```
import investpy

df = investpy.get_stock_historical_data(stock='AAPL',
                                        country='United States',
```

(continues on next page)

(continued from previous page)

```
                                        from_date='01/01/2010',
                                        to_date='01/01/2020')
print(df.head())

             Open   High    Low  Close     Volume Currency
Date
2010-01-04  30.49  30.64  30.34  30.57  123432176      USD
2010-01-05  30.66  30.80  30.46  30.63  150476160      USD
2010-01-06  30.63  30.75  30.11  30.14  138039728      USD
2010-01-07  30.25  30.29  29.86  30.08  119282440      USD
2010-01-08  30.04  30.29  29.87  30.28  111969192      USD
```

3.1.2 Fund Data Retrieval

```
import investpy

df = investpy.get_fund_recent_data(fund='bbva plan multiactivo moderado pp',
                                   country='spain')
print(df.head())

              Open   High    Low  Close Currency
Date
2019-08-13  1.110  1.110  1.110  1.110      EUR
2019-08-16  1.109  1.109  1.109  1.109      EUR
2019-08-19  1.114  1.114  1.114  1.114      EUR
2019-08-20  1.112  1.112  1.112  1.112      EUR
2019-08-21  1.115  1.115  1.115  1.115      EUR
```

```
import investpy

df = investpy.get_fund_historical_data(fund='bbva plan multiactivo moderado pp',
                                       country='spain',
                                       from_date='01/01/2010',
                                       to_date='01/01/2019')
print(df.head())

              Open   High    Low  Close Currency
Date
2018-02-15  1.105  1.105  1.105  1.105      EUR
2018-02-16  1.113  1.113  1.113  1.113      EUR
2018-02-17  1.113  1.113  1.113  1.113      EUR
2018-02-18  1.113  1.113  1.113  1.113      EUR
2018-02-19  1.111  1.111  1.111  1.111      EUR
```

3.1.3 ETF Data Retrieval

```
import investpy

df = investpy.get_etf_recent_data(etf='bbva accion dj eurostoxx 50',
                                  country='spain')
print(df.head())
```

(continues on next page)

(continued from previous page)

```
              Open     High      Low    Close Currency
Date
2019-08-13  33.115  33.780  32.985  33.585      EUR
2019-08-14  33.335  33.335  32.880  32.905      EUR
2019-08-15  32.790  32.925  32.455  32.845      EUR
2019-08-16  33.115  33.200  33.115  33.305      EUR
2019-08-19  33.605  33.735  33.490  33.685      EUR
```

```
import investpy

df = investpy.get_etf_historical_data(etf='bbva accion dj eurostoxx 50',
                                      country='spain',
                                      from_date='01/01/2018',
                                      to_date='01/01/2019')
print(df.head())
```

```
              Open    High     Low   Close Currency
Date
2011-12-07  23.70  23.70  23.70  23.62      EUR
2011-12-08  23.53  23.60  23.15  23.04      EUR
2011-12-09  23.36  23.60  23.36  23.62      EUR
2011-12-12  23.15  23.26  23.00  22.88      EUR
2011-12-13  22.88  22.88  22.88  22.80      EUR
```

3.1.4 Index Data Retrieval

```
import investpy

df = investpy.get_index_recent_data(index='ibex 35',
                                    country='spain')
print(df.head())
```

```
                Open      High       Low     Close   Volume Currency
Date
2019-08-26  12604.7  12646.3  12510.4  12621.3  4770000      EUR
2019-08-27  12618.3  12723.3  12593.6  12683.8  8230000      EUR
2019-08-28  12657.2  12697.2  12585.1  12642.5  7300000      EUR
2019-08-29  12637.2  12806.6  12633.8  12806.6  5650000      EUR
2019-08-30  12767.6  12905.9  12756.9  12821.6  6040000      EUR
```

```
import investpy

df = investpy.get_index_historical_data(index='ibex 35',
                                        country='spain',
                                        from_date='01/01/2018',
                                        to_date='01/01/2019')
print(df.head())
```

```
                Open      High       Low     Close    Volume Currency
Date
2018-01-02  15128.2  15136.7  14996.6  15096.8  10340000      EUR
2018-01-03  15145.0  15186.9  15091.9  15106.9  12800000      EUR
2018-01-04  15105.5  15368.7  15103.7  15368.7  17070000      EUR
2018-01-05  15353.9  15407.5  15348.6  15398.9  11180000      EUR
```

(continues on next page)

(continued from previous page)

```
2018-01-08  15437.1  15448.7  15344.0  15373.3  12890000      EUR
```

3.1.5 Currency Crosses Data Retrieval

```
import investpy

df = investpy.get_currency_cross_recent_data(currency_cross='EUR/USD')
print(df.head())

              Open    High     Low   Close  Volume Currency
Date
2019-08-27  1.1101  1.1116  1.1084  1.1091       0      USD
2019-08-28  1.1090  1.1099  1.1072  1.1078       0      USD
2019-08-29  1.1078  1.1093  1.1042  1.1057       0      USD
2019-08-30  1.1058  1.1062  1.0963  1.0991       0      USD
2019-09-02  1.0990  1.1000  1.0958  1.0968       0      USD
```

```
import investpy

df = investpy.get_currency_cross_historical_data(currency_cross='EUR/USD',
                                                 from_date='01/01/2018',
                                                 to_date='01/01/2019')
print(df.head())

             Open    High     Low   Close  Volume Currency
Date
2018-01-01  1.2003  1.2014  1.1995  1.2010       0      USD
2018-01-02  1.2013  1.2084  1.2003  1.2059       0      USD
2018-01-03  1.2058  1.2070  1.2001  1.2014       0      USD
2018-01-04  1.2015  1.2090  1.2004  1.2068       0      USD
2018-01-05  1.2068  1.2085  1.2021  1.2030       0      USD
```

3.2 Additional Data

As Investing.com provides more data besides the historical one, some of that additional data can be fetched via investpy. Currently, as the package is under-development, some additional functions have been created in order to retrieve more data as indexed in Investing.com.

3.2.1 Stock Company Profile Retrieval

```
import investpy

company_profile = investpy.get_stock_company_profile(stock='bbva',
                                                     country='spain')
print(company_profile)

{
    "url": "https://www.investing.com/equities/bbva-company-profile",
```

(continues on next page)

(continued from previous page)

```
    "description": "Banco Bilbao Vizcaya Argentaria, S.A. (BBVA) is a diversified␣
↪financial company engaged in retail banking ..."
}
```

3.2.2 Fund Information Retrieval

```
import investpy

fund_information = investpy.get_fund_information(fund='bbva plan multiactivo moderado␣
↪pp',
                                                 country='spain',
                                                 as_json=True)
print(fund_information)

{
    'Fund Name': 'Bbva Plan Multiactivo Moderado Pp',
    'Rating': 4,
    '1-Year Change': '-1,19%',
    'Previous Close': '1.103',
    'Risk Rating': 1,
    'TTM Yield': '0%',
    'ROE': '14,02%',
    'Issuer': 'BBVA Pensiones EGFP',
    'Turnover': None,
    'ROA': '4,97%',
    'Inception Date': '16/10/2012',
    'Total Assets': 1670000000,
    'Expenses': None,
    'Min Investment': 30,
    'Market Cap': 34820000000,
    'Category': 'Mixtos Euros Moderados PP'
}
```

CHAPTER 4

Models

4.1 Data Model

As the retrieved historical data is common to every financial product that investpy extracts data from, only a model class has been created in order to store the day-a-day historical data.

So in we define a model in where every value corresponds to each value of the OHLC (Open-High-Low-Close) nomenclature (except on stocks, that it also includes the volume) and it looks like:

```
def __init__(self, date_, open_, high_, low_, close_, volume_, currency_):
    self.date = date_
    self.open = open_
    self.high = high_
    self.low = low_
    self.close = close_
    self.volume = volume_
    self.currency_ = currency_
```

As their names indicate, OHLC values refer to opening, highest, lowest and closing values of the market on a trading day, respectively. And the volume value refers to the number of shares traded in a security day.

Note: The Data model is not usable as it is just a class used for the inner package, transparent to the user. It is used in order to categorize each retrieved value from Investing and then to define its structure and, so on, the structure that either the resulting pandas.DataFrame or JSON file will be based on.

4.2 Search Model

TODO

CHAPTER 5

Stocks/Equities

A **stock** (also known as "shares" or "equities") is a type of security that signifies proportionate ownership in the issuing corporation. This entitles the stockholder to that proportion of the corporation's assets and earnings.

Stocks are bought and sold predominantly on stock exchanges, though there can be private sales as well, and are the foundation of nearly every portfolio. These transactions have to conform to government regulations which are meant to protect investors from fraudulent practices. Historically, they have outperformed most other investments over the long run. These investments can be purchased from most online stock brokers.

Source: *Investopedia*

5.1 Getting Started

To get started using investpy you first need to install it as described on *Installation*. Once you have it installed you can proceed to use it in order to retrieve data from stocks, after importing the package as it follows:

```
import investpy
```

5.1.1 Listing

investpy offers some listing functions that allow the user to get the general information of the indexed stocks on Investing as that information is already stored on CSV files generated automatically on the package installation.

We can either retrieve the whole `pandas.DataFrame` containing all the information stored on the CSV file or a `list` containing just the symbols of the stocks, which are the input parameters for the data retrieval functions.

Also there is a param called `country` which by default is None, which means that the stock listing to be retrieved will include all the available countries (indexed in Investing.com); on the contrary, if the param `country` is an available country, the returned stock information will be filtered by country.

Tip: To get a listing of all the available countries you can use the function `investpy.get_stock_countries()` which will return a `list` containing all the available country names which have stocks

as indexed on Investing.com.

```
# Retrieve all available stocks information as a pandas.DataFrame
stocks_df = investpy.get_stocks(country=None)
# Retrieve a listing of all the available stock symbols
stocks_list = investpy.get_stocks_list(country=None)
```

5.1.2 Recent & Historical Data

The main functions of investpy are focused on historical data extraction, stocks in this case. As the main functionality of the package is to retrieve data from Investing.com, so on, some functions have been developed in order to retrieve both recent and historical data.

As to explain its usage an example is proposed to present historical data retrieval functions:

```
# Retrieves the recent data of BBVA (last month) a spanish stock, as a pandas.
↪DataFrame on ascending order
df = investpy.get_stock_recent_data(stock='bbva', country='spain', as_json=False,␣
↪order='ascending')

# Retrieves the historical data of BBVA, a spanish stock, on the specified date range␣
↪as a pandas.DataFrame on ascending order
df = investpy.get_stock_historical_data(stock='bbva', country='spain', from_date='01/
↪01/2018', to_date='01/01/2019', as_json=False, order='ascending')
```

As we already saw, both functions take some parameters, but some of them are *optional*, which means that the function does not need the user to specify them as they already have a default value.

Both parameters `stock` and `country` are mandatory, since they are the ones that specify which information should be retrieved from Investing.com. Consider that both parameters should match, which means that the symbols of the stock should be an stock from the specified country, if the stock is not found on the specified country, an error will be raised.

When retrieving recent data from an stock, we can additionally specify if we want the output as a json object or not, by setting the parameter `as_json` as either True or False, respectively. We can also set the `order` we want the returned object to have based on dates, where ascending goes from the very first date retrieved until now, and descending goes the other way.

Furthermore, when it comes to historical data retrieval, we also need to specify both `from_date` and `to_date` values, as they are mandatory. Both date values are `str` formatted as *dd/mm/yyyy*.

Tip: If you are not familiar with stocks you can either retrieve a listing of the ones available or check the one presented in Investing Equities.

5.1.3 Company Profile

As an extra feature, via investpy you can retrieve the company profile from a company in order to either classify or analyse them based on the information these companies publicly provide, as it is a self-made description of the company.

```
investpy.get_stock_company_profile(stock='bbva', country='spain', language='english')
```

As explained before, when it comes to data retrieval, both `stock` and `country` parameters are mandatory, and should match; as the default value for the `language` of the retrieved company profile is *english* (as Investing provides company profiles written in english), but besides that, the function also retrieves the company profile on *spanish* from Bolsa de Madrid, which is the additional resource used along this package.

Warning: This function is just available for spanish stocks, since investpy was first created just for Spanish Stocks, Funds and ETFs retrieval. Future coverage for world stocks company profiles is intended, but currently just the spanish ones are available.

5.2 Samples

As the generated dataset has been uploaded to Kaggle some kernels with samples on retrieved data usage have been created by the community.

CHAPTER 6

Funds

A fund is a pool of money that is allocated for a specific purpose. A fund can be established for any purpose whatsoever, whether it is a city government setting aside money to build a new civic center, a college setting aside money to award a scholarship, or an insurance company setting aside money to pay its customers' claims.

A fund is a pool of money set aside for a specific purpose, those pools can are often invested and professionally managed and some common types of funds include pension funds, insurance funds, foundations, and endowments.

Individuals, businesses, and governments all use funds to set aside money. Individuals might establish an emergency fund or rainy-day fund to pay for unforeseen expenses or a trust fund to set aside money for a specific person.

Source: *Investopedia*

6.1 Getting Started

To get started using investpy you first need to install it as described on *Installation*. Once you have it installed you can proceed to use it in order to retrieve data from funds, after importing the package as it follows:

```
import investpy
```

6.1.1 Listing

investpy offers some listing functions that allow the user to get the general information of the indexed funds on Investing as that information is already stored on CSV files generated automatically on the package installation.

The user can either retrieve the whole `pandas.DataFrame` containing all the information stored on the CSV file, a `list` containing just the names of the funds, which are the input parameters for the data retrieval functions; or as a `dict` with all the available fields of information from the funds.

Also there is a param called `country` which by default is None, which means that the fund listing to be retrieved will include all the available countries (indexed in Investing.com); on the contrary, if the param `country` is an available country, the returned fund information will be filtered by country.

Tip: To get a listing of all the available countries you can use the function `investpy.get_fund_countries()` which will return a `list` containing all the available country names which have funds as indexed on Investing.com.

```
# Retrieve all available funds information as a pandas.DataFrame
funds_df = investpy.get_funds(country=None)
# Retrieve a listing of all the available fund names
funds_list = investpy.get_funds_list(country=None)
# Retrieve a dictionary with all the funds and all of their information fields
funds_dict = investpy.get_funds_dict(country=None)
```

Note: The funds `pandas.DataFrame` contains internal package information that is useless for users, but it is provided anyways.

Since the data retrieval functions need both the fund name and the country from where that fund is, there is a function to do so in order to let the user know which are the available countries and, so on, the available funds in those countries. The functions presented below: *investpy.get_funds*, *investpy.get_funds_list* and *investpy.get_funds_dict* have one optional parameter which is the country name so to retrieve just the `pandas.DataFrame`, `list` or `dict` from all the available funds from the introduced country, respectively.

Anyways, before applying that filter, the use of the function *investpy.get_fund_countries* is proposed in order to retrieve all the available countries which have funds.

```
countries = investpy.get_fund_countries()

# Check if a country is either or not in the list & then get all the available funds␣
↪from that country
if 'spain' in countries:
    funds = investpy.get_funds_list(country='spain')
```

So on, every country listed on the previous listing can be used for filtering funds. Note that the country param is needed in data retrieval functions since more than one fund can share the same name but not in the same country.

6.1.2 Fund Search

Before proceeding with the data retrieval functions an additional function is presented, since sometimes the user does not have all the information for the fund to retrieve information from, so on, there is a function which allows the user to search for funds with the specified value for the specified column/field. This function will return a *pandas.DataFrame* with all the results found if they were found, if not, a *RuntimeError* will be raised.

Since the returned object is a *pandas.DataFrame* in the following example both the function usage and further data handling is presented in order to let the user know hos to use the results of the search on the data retrieval functions in order to make it more easy to use. Note that you can either select the value you are searching from the

```
search_result = investpy.search_funds(by='name', value='bbva')

# Get both name and country via pandas.DataFrame index
index = 0
name = search_result.loc[index, 'name']
country = search_result.loc[index, 'country']

# Get both name and country via unique field such as isin
isin = 'ES0113211835'
```

(continues on next page)

(continued from previous page)

```
name = search_result.loc[(search_result['isin'].str == isin).idxmax(), 'name']
country = search_result.loc[(search_result['isin'].str == isin).idxmax(), 'country']

# Or get it manually via printing the resulting pandas.DataFrame
print(search_results)
```

6.1.3 Recent & Historical Data

The main functions of investpy are focused on historical data extraction, and in this concrete case, fund historical data retrieval functions will be explained and sorted out. As the main functionality of the package is to retrieve data from Investing.com and format it so to access it via Python functions, some functions have been developed in order to retrieve both recent and historical data.

As to explain its usage an example is proposed to explain how does historical data retrieval functions work:

```
# Retrieves last month's data of 'Bankia Cauto Pp', which is a fund from 'Spain', as␣
↪a pandas.DataFrame
df = investpy.get_fund_recent_data(fund='Bankia Cauto Pp', country='spain')

# Retrieves historical data of 'Bankia Cauto Pp', which is a fund from 'Spain', on␣
↪the specified date range as a pandas.DataFrame
df = investpy.get_fund_historical_data(fund='Bankia Cauto Pp', country='spain', from_
↪date='01/01/2018', to_date='01/01/2019')
```

Both functions need some parameters, even though some of them are *optional*, which means that the function does not need the user to specify them as they already have a default value.

Both parameters `fund` and `country` are mandatory, since they are the ones that specify which information should be retrieved from Investing.com. Take into consideration that both parameters should match, which means that the name of the fund should be a fund from the specified country, so if the introduced fund is not found on the specified country, an error will be raised.

When retrieving recent data from a fund, we can additionally specify if we want the output as a json object or not, by setting the parameter `as_json` as either True or False, respectively. We can also set the `order` we want the returned object to have based on dates, where ascending goes from the very first date retrieved until now, and descending goes the other way.

Furthermore, when it comes to historical data retrieval, we also need to specify both `from_date` and `to_date` values, as they are mandatory. Both date values are `str` formatted as *dd/mm/yyyy*.

Tip: If you are not familiar with funds you can either retrieve a `list` of the ones available as provided by investpy or check the listing in Investing Funds.

6.1.4 Fund Information

As an extra feature, via investpy you can retrieve information insights for the specified fund on the specified country. This information is the one related to the introduced fund as indexed by Investing.com which will give the user a wider sight on that concrete fund since values such as risk, rating or category are provided by Investing.com and, so on, by investpy.

Its usage is pretty simple since just the *fund* and the *country* are mandatory parameters, but there is also an additional parameter which is *as_json* that can be either True or False whether the information wants to be returned as a `pandas.DataFrame` or a `json`.

```
# Retrieve information from the introduced fund in the specified country
data = investpy.get_fund_information(fund='Bankia Cauto Pp', country='spain')
```

CHAPTER 7

Additional Information

As this is an open source project it is open to contributions, bug reports, bug fixes, documentation improvements, enhancements and ideas. On Github you can find an open tab of issues where you can contribute by opening new issues if needed or contribute to its solving.

Additionally, you can triage issues on investpy Code Triage where you can both open and solve issues so the package can grow and improve faster as the issues solve relevant bugs, problems or needs of the package.

Feel free to contact package administrator via email.

Note: For further information or any question feel free to contact me via email You can also check my Medium Publication, where I upload posts related to Data Science and to investpy basics on Web Scraping.

CHAPTER 8

Frequent Asked Questions - FAQs

In this section the Frequent Asked Questions are answered, so please read this section before posting a question or openning an issue since duplicates will not be solved or will be referenced to this section. Also, if you think that there are more possible FAQs, consider openning an issue in GitHub so to notify it, since if we all contribute this section can be clear enough so to ease question answering.

8.1 Where can I find the reference of a function and its usage?

Currently the `docs/` are still missing a lot of information, but they can be clear enough so that users can get to know which functions can be used and how. If you feel that any functionallity or feature is not clear enough, please let me know in the issues tab, so that I can explain it properly for newcomers, so that answers are more general and help more users than just the one asking it. Docs can be found at: Documentation

8.2 What do I do if the financial product I am looking for is not indexed in investpy?

As it is known, investpy gathers and retrieves data from Investing.com which is a website that contains a lot of financial information. Since investpy relies on Investing data, some of it may not be available in Investing, which will mean that it will not be available in investpy either. Anyways, it can be an investpy problem while retrieving data, so on, there is a search function (`investpy.search_quotes(text, products, countries, n_results)`) that can be used for searching financial products that are available in Investing but they can not be retrieved using investpy main functions.

8.3 I am having problems while installing the package.

If you followed the Installation Guide, you should be able to use investpy without having any problem, anyways, if you are stuck on it, open an issue at investpy issues tab so to let the developers know which is your problem in order to

solve it as soon as possible. If you were not able to complete the installation, please check that you are running Python 3.5 at least and that you are installing the latest version available, if you are still having problems, open an issue.

8.4 How do I contribute to investpy?

Currently I am not admitting any Pull Request since investpy is under development, and so to keep a clean structure, I will be developing new functionalities until code is clean enough to let newcome contributors help. Anyways, the most effective tool you have in order to contribute to investpy are **issues** where you can give me new ideas or some functionallity you would like to see implemented in investpy. You can also use issues in order to report bugs or problems so to help investpy's development and consistency.

8.5 How do I reference investpy?

Since investpy is an open source Python package, whenever you use it, would be nice from you to mention or comment where does the data comes from. This way, investpy can be spread among more users which will consequently improve package usage since more users can contribute to it due to the increasing reach to newcome developers. A sample reference is presented below:

```
investpy - a Python package for Financial Data Extraction from Investing.com
developed by Álvaro Bartolomé del Canto, alvarobartt @ GitHub
```

CHAPTER 9

Disclaimer

This Python Package has been made for research purposes in order to fit a needs that Investing.com does not cover, so this package works like an Application Programming Interface (API) of Investing.com developed in an altruistic way.

Conclude that this package is not related in any way with Investing.com or any dependant company, the only requirement for developing this package was to mention the source where data is retrieved.

CHAPTER 10

Indices and tables

- genindex
- modindex
- search