[1. DESCRIPTION 2](#_Toc23164829)

[2. FILE STRUCTURE 2](#_Toc23164830)

[3. SCRAPER STRUCTURE 2](#_Toc23164831)

[3.1 Bank\_Name.py 3](#_Toc23164832)

[3.2 DataUtils.py 3](#_Toc23164833)

[3.3 fileUtils.py 3](#_Toc23164834)

[3.4 link\_tracker.py 4](#_Toc23164835)

[3.5 TopCompare loan Scraper.py 4](#_Toc23164836)

[4. INSTALLATION AND LAUNCH 4](#_Toc23164837)

[4.1 INSTALLATION 4](#_Toc23164838)

[4.1.1 Parse data. 4](#_Toc23164839)

[4.1.2 Send http requests. 4](#_Toc23164840)

[4.1.3 Navigate file Structure and Mail notification 5](#_Toc23164841)

[4.2 LAUNCH 5](#_Toc23164842)

[5. EXCEPTIONS AND DEBUG 5](#_Toc23164843)

# DESCRIPTION

**TopCompare Scraping Tool** is a data extraction, monitoring and notification tool written in python designed for TopCompare internal use to monitor the interest’s rates of bank partners.

The scaper extracts rates data from bank providers web sites, keep track of changes over time and send notification of changes to a mail list. The extracted rates are stored as csv file in a well-defined file structure.

# FILE STRUCTURE

All the csv files are stored in a bank folder, this folder is structured as shown on the graph

File structure

**History folder**: contains all the **different** scraped rates file over time

**Update folder**: contains all the updated bank rates as csv

The current rates are in the root of the bank folder

**Principle**:

In case of recorded change:

* the current rates file is moved to the history folder,
* a file containing the updated line is generated into the Update folder
* and a new file is created with the daily extracted data as current rates.

# SCRAPER STRUCTURE

to be able to achieve this task the script is divided into 4 different python files:

* *Bank\_Name.py*
* *DataUtils.py*
* *fileUtils.py*
* *link\_tracker.py*
* *TopCompare loan Scraper.py*

## Bank\_Name.py

The file contains all the data extracting procedure for each bank, the main procedures here are:

* A procedure to request for data on the website (usually named ***requesDataFor*** or ***makeRequestFor)***. Depending on the data source which might be a pdf-file, a script or json file through a simulator, this procedure will request the website and return a data set.
* A procedure to format the data (usually named ***bankData***), the data need to be formatted with the appropriate header and type in order to exportable as csv file, this procedure will handle the formatting group them if necessary and return an exportable data-matrix.

**NB***: this procedure differs from one bank to another mainly because of the difference in dataset from each provider*

* A procedure to realize the scraping (named with the suffix ***Scraper***), using others generic procedures defined in other file this procedure will perform the data monitoring and file rearrangement.

## DataUtils.py

All the generic method used to format data are defined in this file, the procedures are used across each bank scraper If needed. The main method here are used for grouping the data and adding some attributes to the data in other to make them easy to handle or facilitate the export procedure.

The ***Group\_data()*** function group all the data with the same rate and term in other to present a range of rate rather than a wide list of individual data. This might be ambiguous during interpretation but the right way of analyzing is the function will group all the data and create a range from the minimal value in the data to the max value although some missing values might be present in the range.

## fileUtils.py

this file defines the generic methods for handling file manipulation and notification sending. It performs

* file verification and creation through procedures like ***checkToCreate(), computeFileName() or createUpdate().***
* File and frame comparison
* Frame import and export
* Update verifications and actions with the ***UpToDate ()*** procedure
* Notification procedure with the ***send\_mail\_to()*** procedure
* And some useful procedures for debugging e.g.

***no\_double***() : a functional expression that returns list of element without duplicates

***show\_double():*** shows all the duplicates in a list of elements in parameter

## link\_tracker.py

The scraper also implements a script for checking link status on TopCompare website, it makes use of request to query links on the web site and record response status. This also generates messages in case of inappropriate response.

## TopCompare loan Scraper.py

This script realizes the scraping for all banks and builds the message for notifications, it executes the Scrape procedure of all bank and stores the potential messages into a variable to be joined to notification mail.

This file implements one procedure ***tcLoanScrape().***

# INSTALLATION AND LAUNCH

## INSTALLATION

The scraper is written in python in version 3 and therefore requires the module python3 to be installed in other to execute it.

The scraper also uses internal and external libraries for various purposes which are native for the first ones and installable for the others. The various aims of those libraries includes.

### Parse data.

The main library required for data parsing are

* ***Pandas*** to create exportable data frame to import and export data from file.
* ***Tabulate*** to display the data in table like format in the console
* ***BS4*** to parse HTML document

These libraries need to be installed from the terminal in the project folder with the command

***Pip install <library>***

### Send http requests.

All the data collected by the scraper are done through HTTP-GET or HTTP-POST requests the required libraries for request are:

* ***Requests*** for data request
* ***Tabula*** for web file access (pdf reader)

The installation is achieved with command ***pip install***

### Navigate file Structure and Mail notification

In other to access OS folders and files the scraper makes use of some internal libraries which are all native in python and can be callable via a simple import these are:

* ***OS*** provides functions to create, move, delete and navigate OS file and folders
* ***GLOB*** useful for time stamping of file
* ***NTHPATH*** used to compute file names

For E-mail notifications some native libraries callable via an import are used, the libraries are:

* ***Email.mime.multipart***
* ***Email.mime.text***
* ***Email.mime.base***

## LAUNCH

The scraper needs to be executed through the ***TopCompare Loan Scraper.py file*** it imports all the other files and implements the notification procedure. Every additional file developed needs to be added in this file for it to be considered during the scraping procedure.

A function ran outside the scope of the main function in the file will make all the OS file operations but the changes be notified.

From the terminal to run the command:

***Python3 /<absolute>/<folder path>/ TopCompare Loan Scraper.py***

Where **absolute folder path** is the actual absolute path to the directory.

# EXCEPTIONS AND DEBUG

Sometimes it might happen that the scraper raises some exceptions, this will likely occur if the data structure has been modified by the provider, or if the web site is down. therefore, some request will not longer be supported by the provider web site. Hence generating the exceptions.

To handle this the following steps can be followed.

* Check internet connection.
* Check if the request headers are still valid.
* Check the response of the web site.
* Check the occurrence of an unsupported request.
* Check the data structure.

For every provider the data handling procedure is the ***bank\_data()*** procedure, in case of reprocessing the data note that data need to have a particular structure in other to be exportable as a frame.