# PROFIT

Python-Based Return On Investment and Financial Investigation Tool

## 1 License



## 2 Usage:

a) Run *PROFIT_main.py* with a Python interpreter. Check the console output. Some PDFs, with data based on the provided examples, are created in the plots-folder. Marvel at the results.

b) Move the provided examples to the *accounts* and *investments* folders, and modify them according to your situation. These files must adhere to certain standards, which are outlined in sec. 4 and 5 below. Change the folders from the example-folders in *PROFIT_main.py*, such that your files are analyzed (ACCOUNT_FOLDER and INVESTMENT_FOLDER).

c) Modify the configurable settings in *PROFIT_main.py* according to your demands.

d) Enjoy the tool and contribute on *GitHub*: https://github.com/MauererM/profit

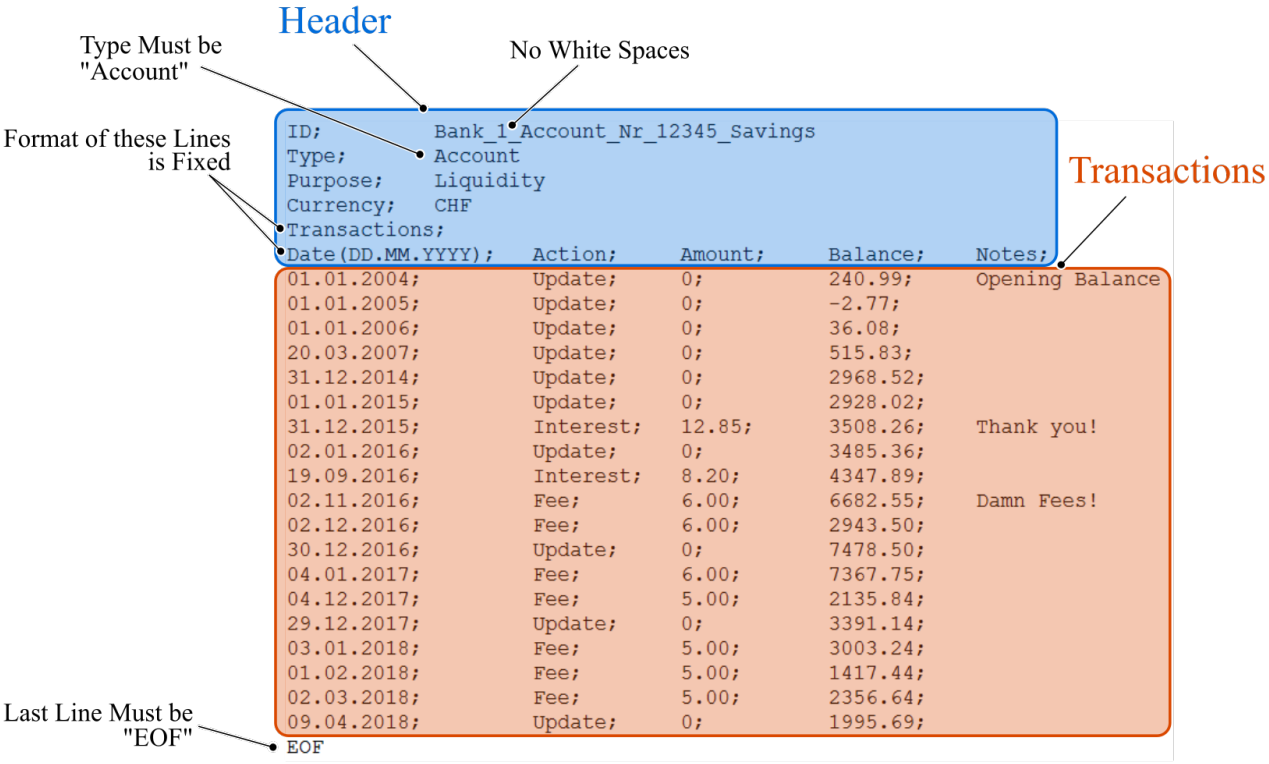
## 3 Key Principles

PROFIT adheres to certain conventions and concepts as follows.

- Assets are divided into accounts and investments
  - **Accounts** simply hold money. Their returns are not tracked, as the maintenance of the inflows/outflows would be cumbersome (Accounts are often used daily, e.g., for paying bills). Account fees and interest can be recorded and visualized.
  - **Investments** are linked to prices that define their values. They are often traded (e.g., stocks). Every in- and outflow of an investment (e.g., buy/sell) is tracked. Cost and payouts are also recorded for a comprehensive data set.
- Values are given for specific dates. A calendar-**day** hence comprises the granularity of the data.
- Data is always analyzed from *today* (when the script is executed) a number of days (user-configurable) into the past. This is the *analysis range.*
  If online data of certain investments are not obtainable automatically, they must be provided as an "update"-transaction that defines the investment's price of today (see ch. 5 below), or the price of the investment must be given as a file (with the correct file name) in the marketdata-folder (see ch. 6 below). A warning will be issued if the necessary data is missing, and the holding period return cannot be calculated.
- All values are converted to the base currency (defined in *PROFIT_main.py*), and then analyzed.
- Code should be long-term maintainable; As few python packages as possible are used, and most data manipulations are performed with native (built-in) python commands and data types.

# 4 Account Files

This is an example of an account file (store as .txt in the accounts folder):




Note the separator (semicolon) between the different columns. This is required for parsing the file. Any white space is stripped when the file is parsed, hence the account ID cannot contain white spaces, for example.

The **header** must be structured as illustrated and as follows:

- ***ID***
  User-definable string that identifies the account, e.g., name of the bank or the account number etc.
- ***Type***
  Must be "*Account*"
- ***Purpose***
  User-configurable, for grouping different assets. This string must be in the list of possible asset-purposes in the *PROFIT_main.py* file (ASSET_PURPOSES)
- ***Currency***
  Encodes the currency. This symbol is also used to automatically obtain exchange rates. Adhere to the ISO 4217 standard of currency designations: https://en.wikipedia.org/wiki/ISO_4217
- ***Transactions***
  This is a placeholder that signals the end of the user-configurable header.
  The following line is the header of the transactions, and its format is also fixed.

The **header-line of the transactions-section** contains the following columns:

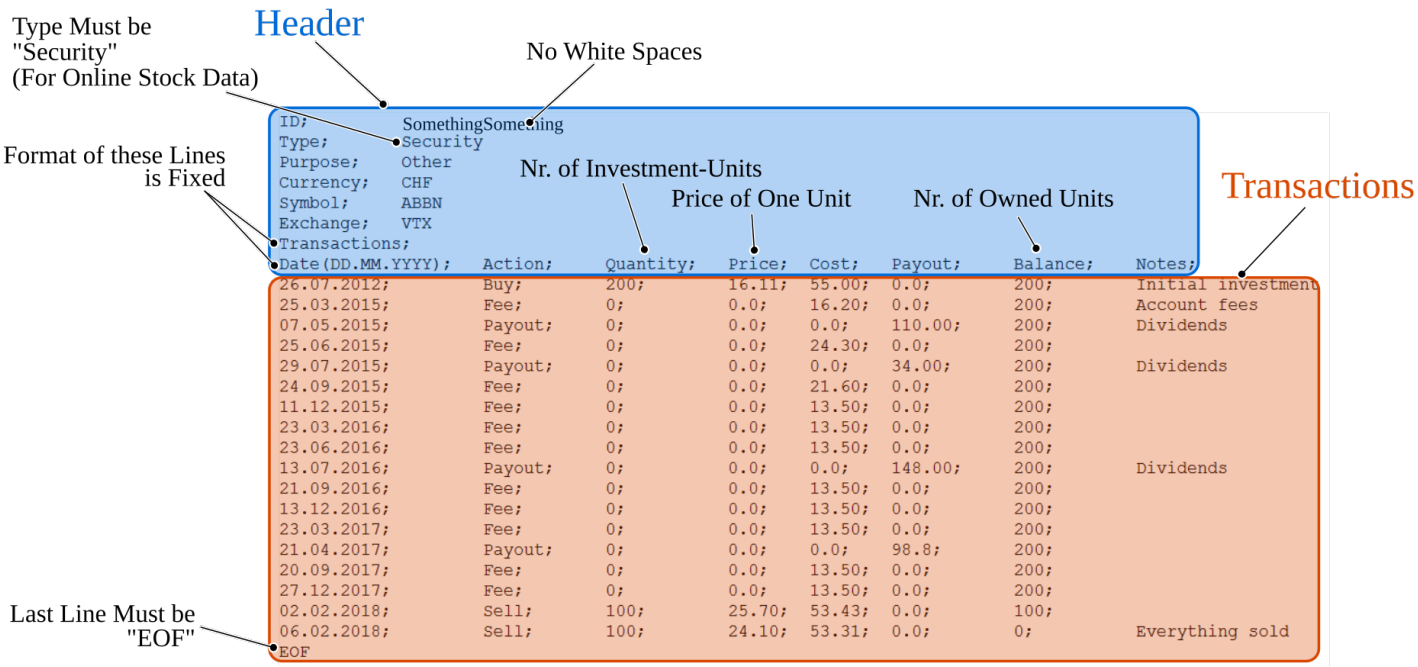
- ***Date (DD.MM.YYYY)***
  Date of a recorded transaction.
- ***Action***
  Encodes the type of transaction. Allowed are the following strings (encoded in *setup.py*):
  - *Update*: Provides a new account-balance. *Amount* must be zero.
  - *Interest*: Payout (e.g., interests). The amount must be given in the "*amount*" column. A new balance can also be provided with the same transaction.
  - *Fee:* Similar than interest, but considers cost associated with the account.
- ***Amount***
  The amount of payouts or cost, in the account's currency

- ***Balance***
  The balance of the account after the transaction, in the account's currency
- ***Notes***
  User-definable string for personal notes.

The last line of the transactions must be the string "EOF" (end of file).

# 5 Investment Files

This is an example of an investment file (store as .txt in the investments folder):




Note the separator (semicolon) between the different columns. This is required for parsing the file. Any white space is stripped when the file is parsed, hence the ID cannot contain white spaces, for example.

The **header** must be structured as illustrated and as follows:

- ***ID***
  User-definable string that identifies the investment, e.g., name of the bank, ISIN number etc.
- ***Type***
  For automatic market price retrieval, the type must be "*Security*". Any other string can be used, but it will not be attempted to obtain market prices online.
- ***Purpose***
  User-configurable, for grouping different assets. This string must be in the list of possible asset-purposes in the *PROFIT_main.py* file (*ASSET_PURPOSES*)
- ***Currency***
  Encodes the currency. This symbol is also used to automatically obtain exchange rates. Adhere to the ISO 4217 standard of currency designations: https://en.wikipedia.org/wiki/ISO_4217
- ***Symbol***
  Ticker symbol of listed assets, e.g., "AAPL", "TSLA" etc. This string is used by the online data retrieval tools to identify the asset.
- ***Exchange***
  Name of the stock exchange, where the investment is traded, e.g., "SWX", "NASDAQ" etc. This string is used by the online data retrieval tools to identify the exchange.
- ***Transactions***
  This is a placeholder that signals the end of the user-configurable header.
  The following line is the header of the transactions, and its format is also fixed.

The **header-line of the transactions-section** contains the following columns:

- ***Date (DD.MM.YYYY)***
  Date of a recorded transaction.
- ***Action***
  Encodes the type of transaction. Allowed are the following strings (encoded in setup.py):

- *Buy*: A certain quantity of the investment (given by the *Quantity*-column) is bought (e.g., a number of stocks). This transaction requires a *quantity* and a *price* (which corresponds to the market price of a single investment-unit). A *cost* can also be given (but no *payout*). The *balance* must correspond to the old balance plus the newly bought units. The first transaction of an investment must be a *buy* action.
- *Payout*: Used to track payouts, e.g., dividends. The amount must be given in the *payout* column. *Price* and *cost* may not be given and *quantity* must also be zero, and the *balance* must be correct. Value given in the investment's currency.
- *Cost*: Similar than *payout*, it considers cost associated with the investment. The value is given in the *cost* column. *Price*, *payout* and *quantity* must be zero, and the *balance* must be correct. Value given in the investment's currency.
- *Sell*: Like *buy*, but the *balance* of the investment is reduced. The value in the *balance* column must correspond to the new balance.
- *Update* (not illustrated in the example above): Used to provide a new *price*, in case online data retrieval is not possible. Simply state the *price* (of one unit of the investment) in the *price* column. *Cost*, *payout* and *quantity* must be zero, but *balance* must be correct.

- ***Quantity***
  Bought or sold investment quantities. Can be a floating-point number.
- ***Price***
  The *price* (in the investment's currency) of a single investment-unit of a *buy* or *sell* transaction. Can also be updated with an *Update*-action (see above).
- ***Cost*** and ***Payout***
  Associated *costs* or yields with the corresponding transactions.
- ***Balance***
  Held investment quantities (e.g., nr. of stocks). Must be correctly updated when *buy* or *sell* transactions are performed.
- ***Notes***
  User-definable string for personal notes.

The last line of the transactions must be the string "EOF" (end of file).

# 6 Marketdata-Files

The folder *marketdata* contains an automatically updated and maintained database of prices of assets, comprising a collection of text files. The files contain two columns; the date and the price of the asset (for one asset-unit).

The specific assets are identified through the file name as follows:

- Traded securities:
  price_<SYMBOL>_<EXCHANGE>_<CURRENCY>.txt
- Foreign exchange rates:
  forex_<CURRENCY>_<BASECURRENCY>.txt

It is possible to manually add and maintain files, if the automatic retrieval of asset prices and exchange rates is not possible.

The idea of this database is to retain data over long periods of time, and to provide an easy to maintain set of offline data.

# 7 Project Structure

| **Name** | **Type** | **Function** |
|---|---|---|
| *PROFIT_main.py* | File | Main script. Run with a Python interpreter to generate the output. |
| accounts | Folder | Stores the files that encode accounts |
| investments | Folder | Stores the files that encode investments |
| marketdata | Folder | Contains historic prices of investments and foreign exchange rates. The files are automatically updated with new data, if possible. Files can be added manually, if prices of some investments cannot be obtained automatically. |
| plots | Folder | The output of the main script is created in this folder |
| doc | Folder | Documentation |

# 8 Used Software

The project requires/is tested and running with the following key python packages:

| **Package / Python** | **Package Version** | **PROFIT Version** | **Purpose** |
|---|---|---|---|
| Python | 3.5.2 | 1.0 | Python |
| googlefinance.client | 1.3.0 | 1.0 | Market data and exchange rates |
| DateTime | 4.2 | 1.0 | Date manipulations |
| pandas | 0.21.0 | 1.0 | Process data from googlefinance.client |
| matplotlib | 2.0.0 | 1.0 | Plotting |

# 9 Revision History

| PROFIT Version | Date | Comment |
|---|---|---|
| 1.0 | 24.04.2018 | Initial running version - Mario Mauerer |
| | | |