PROFIT

Python-Based Return On Investment and Financial Investigation Tool

1 License

MIT License

PROFIT - Python-Based Return on Investment and Financial Investigation Tool

Copyright (c) 2023 Mario Mauerer

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

2 Usage:

- a) Run *profit_main.py* with a Python 3 interpreter. Check the console output. Some PDFs, with data based on the provided examples, are created in the plots-folder.
- 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* (or *config.py*), such that your files are analyzed (ACCOUNT_FOLDER and INVESTMENT_FOLDER).
- c) Modify the configurable settings in *profit_main.py* or *config.py* according to your demands.

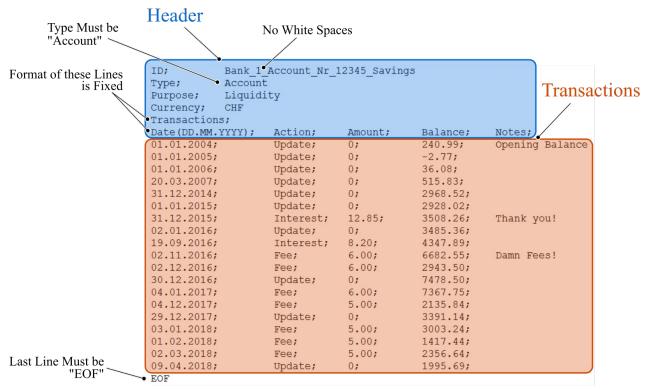
3 Key Principles

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 inand 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
 - "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 in the marketdata_storage 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 or config.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 config.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

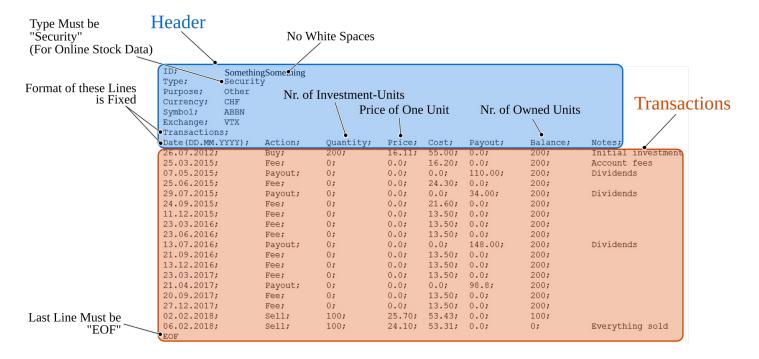
• Notes

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.

Go to yahoo finance and use the ticker as displayed, e.g., ABBN is ABBN.SW. This works best for the alpha vantage API.

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 config.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.
- Split (not illustrated in the example above): Needed when the stock undergoes a split. Simply provide
 the new price and balance. The split ratio will be calculated from the newly provided balance. Cost and
 payout must be empty.
 - Note: The split ratio can also be given in the Quantity column, if the balance is 0. Usually, the ratio is determined/derived from the balance, but if the balance is 0, the ratio can/has to be given in the "quantity" column! Reverse splits (ratio <1) are also possible.

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. The net yield is given by payout – cost, so state the payout value before cost is deducted.

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-Storage Files

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

If a file is missing, it gets automatically created. The file name must adhere to a certain format.

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

The storage entries have priority over online-obtained data, i.e., if online retrieved data and data from these files mismatch, the data from the marketdata-files will be used.

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.

The files are configurable with a header. Some headers are configurable:

Stock/Securities:

```
Header;
ld;<Name>
Split;<date>;<float>
Split;<date>;<float>
Data;
<date>;<value>
...
```

The split-lines are optional. Normal splits have a split-factor >1. Reverse splits are <1.

If a split is present in the storage-header, then the corresponding _provider_ data will be adjusted (and not the stored csv-data!).

CAREFUL: Once the split-data is given in the header, any data from the corresponding provider will always be modified and written to the file (where it then becomes ground-truth).

NOTE: If you introduce or modify a split in a storage-csv-file, you MUST delete all content fo the file (but keep the header!). This forces profit to re-pull all data from the provider, and apply the split going forward.

You also might have to play with the day of the split in the storge-header by +/- 1 day to match the recorded transactions.

Forex- or Index-Files:

Header; ld;<Name> Data; <date>;<value>

7 Project Structure

Name	Туре	Function
profit_main.py	File	Main script. Run with a Python 3 interpreter to generate the output.
profit_src	Folder	Main python package with all PROFIT source code
accounts	Folder	Stores the files that encode accounts
investments	Folder	Stores the files that encode investments
marketdata_storage	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 Revision History

PROFIT Version	Date	Comment	
1.0	24.04.2018	Initial running version - Mario Mauerer	
1.1	13.08.2018	Migrated from (terminated) Google finance API to Alpha Vantage as data provider	
1.2	05.01.2021	Migrated from Alpha Vantage to Yahoo Finance, added/fixed stock splits.	
1.3	Nov. 2023	Improved speed significantly. Fixed some bugs.	
1.4	Nov. 2023	Improved handling of storage- and provider-data. Introduced splits in storage-data as well. Introduced better package-structure.	