# ######## Manual super.py program ########

The super.py is a Python command-line program that can be used for the inventory administration of a supermarket. The database has 2 main .csv files: bought.csv & sold.csv. With the program you can add a bought or sold product to these lists. Furthermore, based on these two .csv files, the program can create 5 types of reports: bought, sold, revenue, profit and inventory.

Below, you will be guided through all the functionalities of the program. Before you start running the example code below in your terminal, first navigate to the folder where you saved the program.

## ## HELP

With the --help command, you get an overview of all the possible input arguments:

python super.py --help

python super.py buy --help

python super.py sell --help

python super.py report --help

## ## DATE

The current date of the program is set in the text file current\_date.txt. Run the following code to check the this:  
python super.py --current-date

To advance the date by a specific amount of days you can use the following code (2 days in this example):

python super.py --advance-days 2

To go back in time, you can use a negative number:

python super.py --advance-days -2

To set a specific date you can use the following code   
(best date to start with for the examples without date specification):

python super.py --set-date 2023-04-08

## ## SUBCOMMAND buy & sell

You can enter a bought or sold product using the code below:

python super.py buy --name orange --price 1.5 --exp 2023-04-05

python super.py buy --name hagelslag --price 2 --exp 2023-05-05 --buy-date 2023-04-20

python super.py sell --name orange --price 2

python super.py sell --name hagelslag --price 4 --sell-date 2023-04-21

For buying you need to add the price it was bought for and the expiration date. For selling you just need to add the price it was sold for. By default, the current date of the program will be used as the sold/bought date of the product. If you bought/sold it on another date, you can add the optional --sell-date command.

## ## SUBCOMMAND report (bought, sold, revenue, profit and inventory)

The program can produce 5 kinds of reports using the following code:

python super.py report bought

python super.py report sold  
python super.py report revenue

python super.py report profit

python super.py report inventory

By default, the program will output tables with data from 2000-01-01 (an arbitrary date before the supermarket started) up to and including the current date of the program. With some optional commands, the program makes the reports based on date from a specific date/time period. Here an example for the bought subcommand (on page 3 of this manual you can also find the same code for the other reports):

python super.py report bought

python super.py report bought --now

python super.py report bought --today

python super.py report bought --yesterday

python super.py report bought --date 2023-01-01

python super.py report bought --timewindow 2023-01-01 2023-04-01

python super.py report bought --month 2023-01

python super.py report bought --year 2022

--now : default of the program, so does not chance the analysis   
--yesterday : report calculated for yesterday

--date : report calculated for a specific date

--timewindow : report calculated for a specific time window

--month : report calculated for a specific month

--year : report calculated for a specific year  
  
Above of the output table that is printed by the program, the date/time period is shown (always called “timewindow” also if it just one day).

The inventory cannot be calculated for a time period. Therefore, only the --now and --date commands work (error message if you use the other commands). When using the --date command, the inventory will be determined on all the data up to and including the specified date. For the inventory report, 2 additional functionalities are available: the final summary of the inventory can be saved as a table (.csv) or as a bar graph (.png):

python super.py report inventory --save-inv-tab

python super.py report inventory --save-inv-bar

This also works for specific date:

python super.py report inventory --save-inv-tab --date 2023-05-01

python super.py report inventory --save-inv-bar --date 2023-05-01

## 

## ## EXAMPLES FOR ALL REPORT TYPES

python super.py report bought

python super.py report bought --now

python super.py report bought --today

python super.py report bought --yesterday

python super.py report bought --date 2023-01-01

python super.py report bought --timewindow 2023-01-01 2023-04-01

python super.py report bought --month 2023-01

python super.py report bought --year 2022

python super.py report sold

python super.py report sold --now

python super.py report sold --today

python super.py report sold --yesterday

python super.py report sold --date 2023-04-07

python super.py report sold --timewindow 2023-03-01 2023-04-05

python super.py report sold --month 2023-02

python super.py report sold --year 2022

python super.py report revenue

python super.py report revenue --now

python super.py report revenue --today

python super.py report revenue --yesterday

python super.py report revenue --date 2023-04-07

python super.py report revenue --timewindow 2023-01-01 2023-05-01

python super.py report revenue --month 2023-04

python super.py report revenue --year 2022

python super.py report profit

python super.py report profit --now

python super.py report profit --today

python super.py report profit --yesterday

python super.py report profit --date 2023-04-02

python super.py report profit --timewindow 2023-01-01 2023-04-10

python super.py report profit --month 2023-02

python super.py report profit --year 2022

python super.py report inventory

python super.py report inventory --now

python super.py report inventory --date 2023-01-01

python super.py report inventory --save-inv-tab

python super.py report inventory --save-inv-bar

python super.py report inventory --save-inv-tab --date 2023-05-01

python super.py report inventory --save-inv-bar --date 2023-05-01

For "report inventory" --yesterday/ --today/ --timewindow/ --month/ --year does not work because inventory cannot be calculated for a time period. (Error message will appear when you try it)

python super.py report inventory --save-inv-tab

python super.py report inventory --save-inv-tab --date 2023-01-01

python super.py report inventory --save-inv-bar

python super.py report inventory --save-inv-bar --date 2023-01-01

python super.py report inventory --save-inv-bar --date 2023-02-01

python super.py report inventory --save-inv-bar --date 2023-03-01

python super.py report inventory --save-inv-bar --date 2023-04-01

python super.py report inventory --save-inv-bar --date 2023-05-01

python super.py report inventory --save-inv-bar --date 2023-06-01

python super.py report inventory --save-inv-tab

python super.py report inventory --save-inv-tab --date 2023-01-01

python super.py report inventory --save-inv-tab --date 2023-02-01

python super.py report inventory --save-inv-tab --date 2023-03-01

python super.py report inventory --save-inv-tab --date 2023-04-01

python super.py report inventory --save-inv-tab --date 2023-05-01

python super.py report inventory --save-inv-tab --date 2023-06-01

2do:

* Sell product that expires first DONE
* Setting date to specific date DONE
* Buy/sell product on specific day DONE
* Make print function
* Exporting to CVS
* Standard plot function

**Requirements**

**Code**

* Well-structured and documented code, including:
  + Clear and effective variable and function names;
  + Use of comments where the code does not speak for itself;
  + Clear and effective separation of code into separate functions and possibly files.
* Use of modules to the extent that it shows you were able to independently read and understand the documentation, and apply the techniques within:
  + **csv**
  + **argparse**
  + **datetime**, including in particular the date object, strftime

and strptime functions and datetime arithmetic using timedelta.

* Use of external text files (CSV) to read and write data.
* A well-structured and user friendly command-line interface with clear descriptions of each argument in the --help section.
* A text file containing a usage guide aimed with peers as the target audience. The usage guide should include plenty of examples.
* The application must support:
  + Setting and advancing the date that the application perceives as 'today';
  + Recording the buying and selling of products on certain dates;
  + Reporting revenue and profit over specified time periods;
  + Exporting selections of data to CSV files;
  + Two other additional non-trivial features of your choice, for example:
    - The use of an external module [Rich(opens in a new tab)](https://github.com/willmcgugan/rich) to improve the application.
    - The ability to import/export reports from/to formats other than CSV (in addition to CSV)
    - The ability to visualize some statistics using [Matplotlib(opens in a new tab)](https://matplotlib.org/)
    - Another feature that you thought of.

**Report**

* Please include a short, 300-word report that highlights three technical elements of your implementation that you find notable, explain what problem they solve and why you chose to implement it in this way. Include this in your repository as a report.md file.