A well-structured and user friendly command-line interface with clear descriptions of each argument in the --help section.

--help

Krijg een helder overzicht van alle beschrijvingen van elk argument in de –help sectie.

**$ python super.py --help**

usage: SuperPy [options]

Keep track of your inventory.

-----------------------------

- Enter the purchased products.

- Input the products sold.

- Get an overview of the supermarket's inventory.

Produce reports on various kinds of data.

-----------------------------------------

- Reporting revenue over specified time periods.

- Reporting profit over specified time periods.

- Visual reporting of costs, revenue and profit over specified time periods.

positional arguments:

{buy,sell,report,grafic}

buy Enter your purchased products.

sell Enter the products sold.

report Produce reports on various kinds of data.

grafic show data grafic

options:

-h, --help show this help message and exit

--advance-time ADVANCE\_TIME, --advance ADVANCE\_TIME,--at ADVANCE\_TIME

Setting and advancing the date that the application perceives as 'today'. Enter the number of days you want to go back in time.

**$ python super.py buy -–help**

usage: SuperPy [options] buy [-h] --product-name PRODUCT\_NAME --price PRICE --expiration-date EXPIRATION\_DATE

options:

-h, --help show this help message and exit

--product-name PRODUCT\_NAME

Enter the product name of the purchased product.

--price PRICE Enter the purchase price of the purchased product.

--expiration-date EXPIRATION\_DATE

Insert the expiration date as: YYYY-MM-DD

$ python super.py sell –-help

$ python super.py report --help

$ python super.py report inventory --help

$ python super.py report revenue --help

$ python super.py report profit --help

$ python super.py grafic --help

$ python super.py graphic revenue –help

$ python super.py graphic profit --help

$ python super.py graphic costs --help

$ python super.py graphic all\_data --help

**Code**

Be creative with your implementation! We intentionally keep the specification open to encourage you to be creative with this project. However, to obtain a passing grade, you will at least need to satisfy the following requirements:

* 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'.

--advance-time <aantal\_dagen>

Bij eerste keer runnen van een command wordt er automatisch een tekst bestand ‘datum.txt’ aangemaakt met de datum van vandaag erin. Elke nieuwe dag dat er een command wordt gegeven aan de command line tool zal dit tekst bestand automatisch vernieuwen met de huidige datum.

Dat gezegd hebbende, kan je terug gaan in de tijd (bijvoorbeeld als je gisteren je gekochte/verkochte producten bent vergeten in te voeren, of als je een overzicht van de inventaris van een aantal dagen geleden wilt).

Dit doe je door middel van de volgende command;

**--advance-time <aantal\_dagen\_terug\_in\_de\_tijd>**

Vervang de tekst tussen de < > voor het aantal dagen dat je terug in de tijd wilt gaan.

Recording the buying of products on certain dates.

Voer gekochte producten in in de inventaris, door middel van de voldende command:

**Python super.py buy -–product-name <productnaam> --price <inkoopprijs> --expiration-date <houdbaarheidsdatum>**

Na het runnen van bovenstaand command zullen de volgende acties worden genomen:

* Het ingevoerde product zal automatisch een unieke bought-id toegewezen krijgen.
* De buy\_date van het product zal automatisch de datum uit ‘datum.txt’ halen. (indien je dus gister gekochte producten wilt invoeren kan je gebruik maken van command ‘-–advance-time 1’).
* Indien het om 1 uniek product gaat zal het product vervolgens worden toegevoegd aan de inventaris. Indien het product al in de inventaris is zal de inventaris de hoeveelheid van dit product noteren.

Recording the selling of products on certain dates.

Voer verkochte producten in door middel van de volgende command:

**Python super.py sell -–product-name <productnaam> --price <verkoopprijs>**

Na het runnen van bovenstaand command zullen de volgende acties worden genomen:

* Het ingevoerde product zal automatisch een unieke sold-id toegewezen krijgen.
* De sell\_date van het product zal automatisch de datum uit ‘datum.txt’ halen. (indien je dus gister verkochte producten wilt invoeren kan je gebruik maken van command ‘-–advance-time 1’).
* Indien het om 1 uniek product gaat zal het product vervolgens worden verwijderd uit de inventaris. Indien het product meerdere malen in de inventaris is zal de inventaris de hoeveelheid van dit product verminderen.

Reporting revenue and profit over specified time periods.

Vraag de omzet of winst op van een bepaalde periode met 1 van volgende commands:

**Python super.py report revenue –today**

*Om de omzet van vandaag in te zien.*

**Python super.py report revenue –yesterday**

*Om de omzet van gisteren in te zien.*

**Python super.py report revenue –date <YYYY>**

*Om de omzet van een bepaald jaar in te zien.*

**Python super.py report revenue –date <YYYY-MM>**

*Om de omzet van een bepaalde maand in een bepaald jaar in te zien.*

**Python super.py report profit -–today**

*Om de winst van vandaag in te zien.*

**Python super.py report profit –yesterday**

*Om de winst van gisteren in te zien.*

**Python super.py report profit –date <YYYY>**

*Om de winst van een bepaald jaar in te zien.*

**Python super.py report profit –date <YYYY-MM>**

*Om de winst van een bepaalde maand in een bepaald jaar in te zien.*

* + 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](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](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.

* You may use Markdown for your report, but it is not required.
* To assist your explanation you may use code snippets.